Improving the Performance of Wide Area Networks

Thesis

UNRESTRICTED

Improving the Performance of Wide Area Networks

Alan Gene Holt BSc (Hons) Computer Science

May 1999

This thesis is submitted to the Open University for the degree of

Ph.D in the discipline of Telematics

DATE OF SUBMISSION: 29 APRIL 1999
DATE OF AWARD: 16 JULY 1999

Abstract

Research in to the performance of wide area data networks is described in this thesis. A model of wide area network packet delays is developed and used to direct the research in to methods of improving performance.

Wide area networks are slow and expensive compared to the computer systems that rely on them for communication. Typically data networks are packet switched in order to make efficient use of resources. This can lead to contention, and the mechanisms for resolving contention can bring about further delays when demand for resources is high. In this thesis, network users are viewed as interacting decision makers with conflicting interests, and Game Theory is used to analyse the effects users have on each other's performance. It is asserted in this thesis that wide area network performance is an ethical issue as well as a technical one.

Compression is examined as a technique for reducing network traffic load. While load reductions can reduce the time packets spend waiting in buffer queues experimental results show the compression process itself can present a bottleneck if CPU resources are limited.

The other inhibiting factor with regard to wide area network performance is the time it takes for a signal to propagate through a transmission medium. Propagation delays are bounded by the speed of light and becomes significant as the distance between computer systems increases. Mirrors and Caches are methods of bringing data closer to the user, thereby reducing propagation delays and capping traffic loads on long haul communication facilities. The performance benefits of replicating data within a wide area network environment are studied in this thesis.

Acknowledgements

I would like to thank my supervisor John Monk of the Open University for all his help, guidance and constant support. I would also like to thank Richard Wykeham-Martin of Lucent Technologies for his support and for providing the resources needed to carry out this research. The comments and suggestions from David Reed of the Open University were a valuable contribution to this thesis. I am grateful for his time and effort. I am also grateful to Dick Adams of Lucent Technologies for his help with some of the mathematical analysis.

Special thanks are due to my parents for the investment they made in my education. I would not have got this far without it.

Glossary

AFS	Andrew File System
ARIMA	Auto-regressive Intrgrated Moving Average
ATM	Asynchronous Transfer Mode
BIND	Berkeley Internet Name Domain
CPU	Central Processing Unit
DC	Teamwork Data Controller
DCE	Data Circuit-terminating Equipment
DM	Teamwork Display Manager
DNS	Domain Name Service
DTE	Data terminal equipment
FTP	File Transfer Protocol
GIN	Global Intrgrated Network
IBMO	International Bandwidth Management Option
I/O	Input/Output
IPC	InterProcess Communication
ICMP	Internet Control Message Protocol
ISDN	Integrated Services Digital Network

JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LRU	Least Recently Used
MPEG	Motion Picture Experts Group
NCSA	National Center for Supercomputing Applications
NFS	Network File System
PC	Personal Computer
Ping	Packet INternet Groper
PSTN	Public Switched Telephone Network
POTS	Plain Old Telephone Service
PPP	Point-to-Point Protocol
RCP	Remote CoPy
RPC	Remote Procedure Call
TCP/IP	Transmission Control Protocol/Internet Protocol
TFTP	Trivial File Transfer Protocol
URL	Uniform Resource Locator
UUCP	Unix to Unix CoPy
WAN	Wide Area Network
WWW	World Wide Web

Published Work

This section details the work published as a result of this research programme and how it is incorporated into the thesis

The paper "Do Disk Drives Dream of Buffer Cache Hit?" was presented at the ACM conference "Ethics in the Computer Age" in November 1994 and was published in the conference proceedings [107]. In this paper computer systems performance is subjected to ethical analysis. It argues that system performance is affected by user behaviour and is therefore an ethical issue as well as a technical one. This paper was used primarily as introductory material in CHAPTER 1 of this thesis.

The paper "The Network Application from Hell" was published in a chapter of the book "Social and Ethical Effects of the Computer Revolution" in 1996 [108]. This paper discusses the exponential growth of the World Wide Web. The Web's hypertext-based interface provides a better method of coping with navigation of the Internet than the early command line based applications. This together with the Web's expanding array of applications like formatted text, images, audio and video, has led to the explosion in the number of users of the Internet, which in turn affects performance. The paper highlights the dilemma that exists between providing a certain level of perform-

ance and a cost effective service. This paper is also used as introductory material in CHAPTER 1.

The paper "Prisoners, Chicken, Volunteers, Free Riders and Suckers: Dilemmas on the Internet" was published in the IEE Engineering Science and Education Journal, in April 1997 [109]. This paper examines further the "dilemma" raised in the paper "The Network Application from Hell". It uses Game Theory to examine the conflict of self-interest versus group interest within a shared Internet environment. The main body of this paper can be found in CHAPTER 2, Section 2.3 of this thesis.

CHAPTER 1 *Introduction*

This thesis is about improving the performance of wide area networks. Wide area networks are slow and expensive compared with the computer systems that rely on them for communication. In a distributed computing environment wide area networks can present a bottleneck. The time it takes for a signal to propagate through a transmission medium is bounded by the speed of light and becomes significant as the distance between (communicating) computer systems increases.

Furthermore data networks (like the Internet) use packet switching technology in order to make efficient use of resources. This can lead to contention, and the mechanisms for resolving contention can bring about further delays when demand for resources is high.

Technology brings about benefits by enabling people to complete tasks that would be difficult to do manually. Typically, successful completion of a tasks means completing it in a specified time, therefore performance becomes an important issue in determining the rewards gained from using the technology. This is certainly the case for wide area networks.

The ability to share computing resources across national boundaries has been facilitated by advances in wide area networking technology and the distributed computing applications that operate over them [59].

Upgrading the bandwidth of communications facilities can be an expensive (and a common) solution to wide area network performance problems. However, by understanding how users access distributed data, techniques like compression and replication can be exploited to reduce delays and cap traffic volumes on wide area networks.

The research in this thesis focuses on the network technology of the Internet. The Internet evolved from an experimental packet switched network called the ARPAnet which was started in the late 1960s by the US Department of Defence. The Internet came in to being in the mid-1980s with the development of the TCP/IP protocol. The significance of the TCP/IP protocol is that it allows computer systems on different networking technologies to communicate, that is it enables "a homogeneous communication system" on top of "heterogeneous hardware" [59]. The recent rapid growth of the Internet is attributed to the development of the World-Wide Web application in the late-1980s and early-1990s.

The work in this thesis was conducted within Lucent Technologies which is a global company that operates a large *Intranet* based on Internet technology and protocols. Furthermore Lucent uses the World-Wide Web extensively as a means of distributing internal corporate information and as an interface to the external public Internet (as probably do many corporations).

Wide area network performance is investigated by studying the performance of the World-Wide Web. The advantage of studying the Web is that its use is wide-spread, so any improvements to its performance would benefit a large proportion of the Internet community. Furthermore the Web is responsible for much of the traffic on the Internet

(as well as the Lucent Intranet). In which case measures to reduce Web traffic would benefit users of other applications as well.

1.1 The Need for Wide Area Networks

People use computers for many different functions, for example: writing documents, accessing data bases, sending e-mail, and viewing World-Wide Web pages. According to [68] "[i]nformation touches all human activity". Computers bring about commercial and social benefits because they can "manipulate information far faster than humans" [68] and therefore "enable people to be more productive" [14].

While computers do operate in a stand-alone environment, there is a growing trend towards distributed computing, where information resides on storage devices that are located remotely from users' computer systems and are accessed using a network. The rate at which information can be accessed over the network impacts the rate at which users can perform those functions on their computer systems. This in turn affects their productivity.

The total productivity facilitated by a wide area network is a function of the number of users and their demand for information (the rate at which they request data). Any individual user's productivity is limited by two factors; their desire to request data and the wide area network's data rate capabilities.

Some users may only use the network occasionally to send textual messages via e-mail. This sort of usage is unlikely to stretch the data rate capabilities of today's networks. Alternatively some may be frequent users of the World-Wide Web, downloading high resolution graphical images or real time video/audio sequences.

No matter how strong the desire to access information a user's productivity cannot increase without limit. The wide area network itself has a finite data transfer capacity.

Furthermore, any individual user's productivity is diminished by other users sharing the resources.

The emergence of wide area networks has dramatically reduced the cost and increased the speed of global communication [233]. The *first-order* [155] effect of this technology is that people adopt new methods of communication. The *second-order* effect is that people communicate more. The new technology provides a new convenience in communicating such that "people tend to be more free with their information distribution" [232]. This in turn brings about a *third-order* effect in that communication resources become overloaded with network traffic which impacts the technology's ability to fulfil user requirements (in this case the requirement to communicate).

1.2 Growth of the Internet

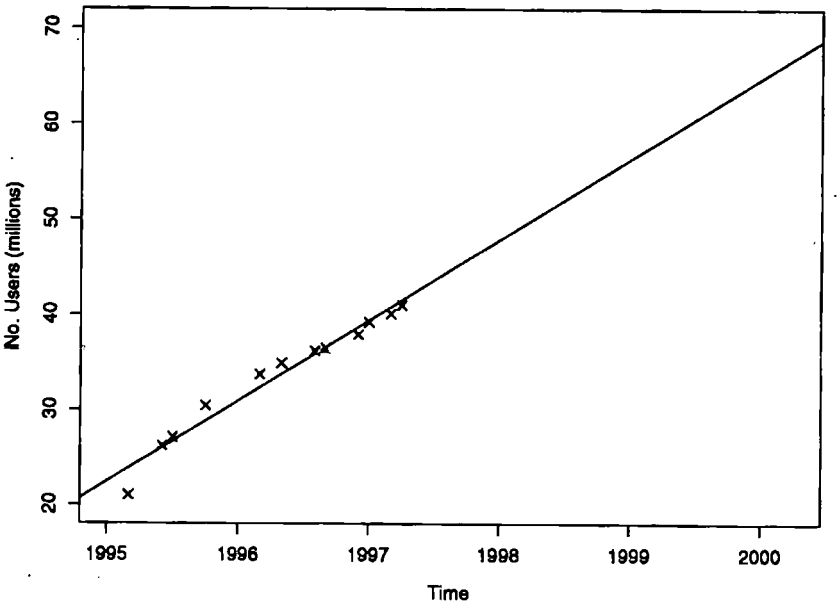
Only a few years ago public awareness of the Internet was fairly limited. The Internet used to be the "communications province of scientists and engineers" [205] and as such its user population was limited. However according to the Economist

> The growth of the Internet has been the most astonishing technological phenomenon of the last decade [72].

Huitema claimed at the beginning of 1994 the number of users on the Internet was 20 million [115]. According to the Economist this number had risen to 50 million by October 1996 [72]. By the end of 1998 Taylor claimed that there were between 60 and 100 million using the Internet [231].

The magazine `.net` publishes an estimate of the number of users on the Internet in each issue. The graph in Figure 1 shows the trend of this growth fitted to data from a number of "back issues".

Figure 1 Growth of users on the internet



The `.net` magazine figures are somewhat lower than that of the Economist and Huitema. According to `.net` magazine the number of users on the Internet in October 1996 was 38 million. The linear regression analysis shows an annual growth of 0.7, in which case 50 million users will be using the Internet by early 1998. By the year 2000 the figure will be around 65 million.

While network access is much more wide spread than it was a few years ago, people that have it are still in the minority, as Wayt says about his Web page

> For every person technologically equipped to read this article, there are about one hundred more that are not [244].

Given that there are many more "potential" network-citizens waiting to come on-line (plus the desire to provide them with access), we should expect to see the trend in Figure 1 to continue and hence traffic volumes to increase.

1.3 Network Applications

Communications technology is progressing rapidly but so too does the computer technology that uses it. The increase in computing power over recent years has meant that applications are no longer limited to text based input and output. Applications can now process video and audio and, as a consequence have greater appetites for bandwidth when they are operated in distributed environments.

According to Dertouzos [68] network speeds have increased by "awesome strides". In the 1980s 10Mb/s local area networks were considered high speed [90]. At the time of writing this thesis 100Mb/s local area networks were becoming common place with gigabit per second being promised in the near future [18].

In the wide area network arena too, gigabit speeds are expected [191] and a number of research institutions [243] have reported advances in fibre optic technology which has enabled them to achieve terabit per second data rates. Nevertheless the economic costs of wide area network links are still quite substantial. Channel capacities of 1-2Mb/s are typically considered high [187]. It may be difficult for wide area networks like the Internet to keep pace with the "burgeoning flow of megabit traffic" [43].

Any successful computer system has its *Killer App.* For the personal computer it was the spreadsheet [42], for the Internet it is the World Wide Web [133]. According to Hughes [113] the Web was the 11th most popular Internet application in terms of network traffic at the end of 1993 when there were 623 servers [91]. Twelve months earlier it was only 127th. At the end of 1994 the number of Web sites was estimated at 12,000 [91]. By 1997 Thomson et. al. [234] reported that the Web was the "single largest Internet application" accounting for 65 to 80 percent of the traffic on one of the international backbone links.

Before the introduction of the Web the Internet was a "cold and forbidding" [133] place. Early Internet search and retrieval tools were primarily command-line based. Berners-Lee, the inventor of the Web, provides the following description of these tools

> Much information is in fact available on-line, but references to it involve complicated instructions regarding host names, logon passwords, terminal types and commands to type, sometimes needing the skilled interpretation of a network "guru" [24].

Despite the wealth of information some members of the user community may have exhibited a reluctance to access it because of the steep learning curve of the tools available at the time, and "not everyone wants to spend months discovering how 20-year-old operating systems work" [133].

The "hard-to-master" nature of early navigation tools had a regulating effect on the amount of requests for information available on the Internet. The Web, however, requires less in the way of "network guru" skills in order navigate and browse the Internet information space, as Hughes states

> The Web offers a very simple-to-use interface to the traditionally hard-to-master resources on the Internet. It is probably this ease of use as well as the popularity of many graphical interfaces to the Web that caused the explosion of the Web traffic in 1993 [113].

The consequence of this is that the scope of Internet citizenship has extended beyond technical domain, resulting in a significant increase in usage of the Internet, with possible detrimental effects on system performance. Experiences of bad performance are wide spread. In a letter that appeared in the magazine *Internet* a user wrote

> When I started using the Internet about 18 months ago, I could connect, flick from page to page on the Web, talk in semi real-time on IRC... Now I'm lucky to get a response. Going from page to page is like watching wet paint dry [75].

Because of applications like the Web, the Internet has become a victim of its own success. Take for instance the collision of the Shoemaker-Levy comet with Jupiter. NASA was able to obtain some of the best pictures from sources like the Hubble telescope and Voyager 2. The Web provided an effective method of distributing high resolution images of event. The comet's popularity and the availability of such images placed a significant load on the SL9 server (and presumably the network). According to Treese [236]

> During the peak hours, the NASA server handled 6000 requests. "The load on our primary server was sufficiently high for us to consider closing down the service," says Syed Towheed, a systems programmer with NASA.

Paradoxically, the applications that enable users to be more productive can have an adverse effect on the underlying system performance, almost to the point of rendering itself unusable.

1.4 User Access Behavior

While advances in technology bring about performance increases in system components (disk drives, memory devices, cables), "good performance" is not necessarily guaranteed. According to Loukides a system that is "used appropriately will give better performance than one that is not" [150].

Not only does the technology affect network performance but also human behavioural factors. Users' actions bring about good (or bad) performance. Loukides' statement is an *ethical* observation. Ethical observations are based on the theories one holds. Harman ([99] pp 3-10) asserts that making an ethical observation regarding some event or action is more than just "seeing" a pattern of light on the retina. In making an ethical observation regarding some event or action, one makes a judgement as to its goodness or badness. In other words one not only "observes" the event or action, one observes

that it is good or bad. Network performance can be considered good or bad and there are various ways in which it can be made better, but not without cost. An analysis of arguments about improvements in network performance is both a technical matter and an ethical issue.

Shared resources are frequently a victim of the "tragedy of the commons" [165] scenario. Motivated merely by maximising their own personal gains, users can over utilise any shared resource. Congested networks are brought about by the following three usage characteristics

- Project Priority
- Intentional misuse
- Unintentional misuse

Project priorities can increase the demands for computer system resources. In the event of organisational crisis, such as an impending dead-line, demand for computer and network resources can cause congestion.

Intentional misuse occurs when users deliberately consume large amounts of resources. This is typically done by experienced users who know how to get the most out of the system. A documented cases of such misuse is the *Internet Worm* [74]. Many computers on the Internet were infected by a Worm program which went out of control. The operation of the Internet was disrupted for many days.

Users may not be aware of the effects of their actions on system performance, and therefore, may create contention problems unintentionally. Commercial organisations tend to expect their employees to use network resources appropriately out of a sense of duty. This goes beyond merely refraining from using the corporate network for personal use. Even when using it for business purposes users are expected to "observe conventional

rules of social behaviour and professional conduct" [232]. Therefore in such an organisation even unintentional misuse may not be acceptable.

The public Internet has guidelines regarding its usage (commonly referred to as "Netiquette"). However these guidelines have been severely challenged [105] which could make any altruistic use of the network difficult to bring about. Lavin makes the following attempt:

> Mindless misuse of Internet bandwidth is seriously slowing down connection speeds... What I say to you all is: stop it now [142]!

However technological environments make it difficult for users to *observe conventional rules of social behaviour.* That is users may not have (or want) the understanding of the underlying technology in order to form the necessary theories from which they could make ethical observations as to appropriate use of the network. That is "we do not blame any man very severely for omitting an action of which as we say, 'he could not be expected to think.'" [172].

In the current environment there is little to prevent users of a network from "stepping on" one another, that is, a "system with a heterogeneous user population has the potential for one set of users to monopolise system resources" [150].

Information systems which have a high degree of openness, the users' ability to "step on" one another is greater than on systems where administrators and system designers implement constraints. There are those that are of the opinion that performance problems on the Internet are due to its pricing structure, because "at present users pay less than the true cost of carrying their messages over the network" [72]. The Internet currently operates two pricing structures [165]:

- Flat-rate
- Capacity-based

With flat-rate pricing users pay a fixed access fee, that is the fee is constant irrespective of usage. With capacity-based pricing users are charged according to the speed at which they access the Internet. Telephone calls are charged by distance and time, and because the telephone industry is regulated in most countries, charges are typically fixed by commissions. The Internet on the other hand is unregulated, and as such its current pricing strategies are "an invitation to use the network as intensively as possible, even if it causes problems for others" [72].

The Usenet bulletin board system has experienced much "misuse" in the form of "Spams". One of the most notorious cases is described in Jannife's "Spam Wars" article [125] about Cantor and Seigal who used the Usenet system to advertise their legal services. The lawyers employed a computer professional to develop a program that would post Usenet articles such that they reached the widest possible audience (by posting to all news groups). The Usenet system enabled them to increase their productivity in terms advertising their services. However the general Usenet population disapproved of this practice and employed various methods of expressing their "moral" outrage. These methods include "flaming" or "mail bombing" the lawyers. Typically such actions have detrimental effects on everyone's performance.

Currently, there is considerable public awareness of the Internet. A few years ago this was not the case and its accessibility was fairly limited. The following statement on Internet Relay Chat (IRC) by Reid reflects on the elitist nature of the Internet in 1991:

> The people who make up the IRC community are effectively preselected by external social structures - access to IRC is restricted to those who have access to the Internet computer network. There are many such people - the Internet spans countries as diverse as Germany, the United States, Japan, Israel, Australia and Korea. However, those individuals who use IRC will be in an economically privileged position in their society. They have access to high technology [204].

Performance improvements can be brought about by usage-sensitive pricing [165], but only for those willing to pay. Usage-sensitive pricing therefore signals a return to an "elitist" access policy (albeit different from that described by Reid [204] in 1991). Consequently this system has its critics. Eichin and Rochlis for example believe that "openness and free flow of information is the whole point of networking" [74].

Furthermore, many people would consider the deconstruction of a society where only a privileged few have access to on-line information resources a good thing, after all "the more people connect to it, the bigger the community of interaction" [96]. This has now become an issue in the political domain. Anne Campbell MP outlines her support for right of network citizenship:

> The "have it all" culture of the '80s left in its wake too many people who had nothing - and in shaping our new digitised world we must ensure that access is open to all, not just the information rich and the access privileged [41].

While usage-sensitive pricing goes against the philosophy upon the which the Internet was developed [37] the author believes that levels of service delay based upon ability to pay will ultimately come about. It is likely that those willing to "pay more to circumvent congestion" [165] will support future investment of network capacity. However there are concerns that this system will result in a "two-tier society of information haves and have-nots" [51] whereby "'poor' users will be deprived of access" [154]. This reinforces the need to for research in to improving the performance of wide area networks. It therefore seems desirable, to deploy methods such as compression and replication in order to drive down the cost of communicating such that a "base level of service" [154] is within economic reach of all users.

1.5 Network Architecture

The PSTN (Public Switched Telephone Network) was primarily designed for the passing of voice traffic but it can also be used for transporting computer data. Essentially there two forms of PSTN communications facility:

- dial-up
- leased line.

Dial-up facilities, either POTS (Plain Old Telephone Service) or ISDN (Integrated Services Digital Network), enables one host computer to "call" another. The establishment of the call sets up a "circuit" through the PSTN and exists until one computer initiates a signal to "tear down" the call. Leased line facilities are dedicated communications circuits through the PSTN and require no call set-up or tear-down (leased line facilities are charged on a fixed rental basis whereas dial-up facilities are charged on call duration).

Information is transferred between host computers by exchanging discrete units of data called messages. Messages can be any form of information and of variable size. Hosts may exchange sequences of messages as part of some larger transaction (called a session [26]). However hosts may not be continually exchanging messages during the course of a session.

Dial-up and leased line facilities are based on *circuit switching* technology. The main advantage of circuit switching is that they provide a constant quality of service because circuits are allocated dedicated resources through the PSTN. For leased lines such resources are allocated for the entire rental period (which could be months or years) whereas for dial-up facilities, resources are only allocated when a call is in progress. Nevertheless these resources are allocated even when no data is being sent. Clearly this is an inefficient use of the network resources.

Also such communications are point-to-point. For multiple hosts to communicate requires a "mesh" of leased lines which can be prohibitively expensive. In order to connect N_H hosts in a fully meshed configuration the number of leased lines required would be

$$\frac{N_H(N_H-1)}{2} \quad (1)$$

Dial-up facilities are a little more flexible in that a computer can communicate with many other computers with only one dial-up line, but not simultaneously.

In a message switching environment [64] hosts do not connect to each other directly (through the PSTN), instead they pass their messages onto an intermediate node (which is co-located at the same site). The intermediate node is connected to intermediate nodes at other sites via the PSTN (typically using leased lines). Messages are *switched* to the appropriate intermediate node which in turn forwards it on to the destination computer. Each intermediate node will serve a number of hosts. If the number intermediate nodes is N_I and $N_I < N_H$ the result is a reduction in the number of communication facilities as

$$\frac{N_I(N_I-1)}{2} < \frac{N_H(N_H-1)}{2} \quad (2)$$

Message switching introduces a certain amount of overhead in that additional information must be transmitted with each message in order to instruct the intermediate node on how the message should be switched. Nevertheless message switching makes more efficient use of network resources than circuit switching because hosts can *share* communications links (through the intermediate nodes).

In a fully meshed circuit switched environment there is no contention, whereas in message switched environment while the transmission of a message is in progress on a communications facility, other messages requiring the same facility will be *blocked* for the

duration of the transmission. Furthermore the message sizes are limited only by the amount of space required for a host to store it, so when messages are large a communications facility can be tied up for a long period of time.

In order to overcome this, the concept of *packet switching* was introduced whereby messages are transferred in packets. The difference between packets and messages is that the maximum packet size is determined by the network, whereas the maximum message size is determined by the sending application. If a message is greater than the maximum packet size, the message is segmented into multiple packets.

Typically, modern data networks are packet switched. Like message switching, packet switching makes more efficient use of communication facilities than circuit switching. Packet switching introduces more overhead (than message switching) because *each* packet of a message must carry switching information for intermediate nodes. The advantage of packet switching is that packets of a message can be transmitted over communication facilities *interleaved* with the packets of other messages. Thus small messages are not delayed for long periods of time by large messages. However the problem of contention in packet switched networks exists just as it does for messages switching. The likelihood of being blocked increases with network traffic (packets) and the amount of traffic is related to the number of users sending messages across.

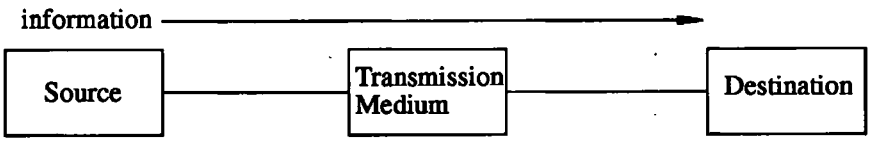
Messages sizes also impact network traffic. The keystrokes of an echoplexed virtual terminal will generate one byte messages, which can be transmitted across the network in a single small packet. Full motion MPEG messages on the other hand could be in the order of gigabytes, the transmission of which would require many packets. So the generation of packets onto the network will further increase as applications grow in sophistication.

The third-order effects of communication give rise to "one of life's disagreeable activities, namely waiting in line" [137]. As traffic intensities increase, packets take longer to traverse the network which impacts performance of applications using the networks. This is very much in evidence on the Internet.

1.6 Moving Data Closer to the User

The need to communicate arises from the need to relay some piece information (a message) that is not known a priori [60]. A communication system (Figure 2) consists of a source (information sender) a destination (information receiver) and transmission medium. The transmission medium may be as simple as an interface cable or as complex as a packet switched network (like the Internet) where there would be multiple sources and multiple destinations.

Figure 2 A communication system



The purpose of the transmission medium is to relay some representation of that information from the source to the destination. That representation will be in the form of a message which, in a computer data network will be a sequence of binary digits.

The speed at which signals travel through a communications medium is bounded by the speed of light, so the greater the distance between source and destination the longer it takes a packet to propagate through the network. Furthermore the economic cost of transferring data over the transmission medium is a function of distance and capacity of the communications facility. As cost is very often a prohibiting factor data rates tend to diminish as source-destination distances increase.

Performance improvements can be brought about by moving data closer to the user [191]. There are a number of established methods of doing this. If a destination was given a *model* of the expected data stream then it could use the model to *predict* some of the information the source intended to send. While data is not entirely predictable, some redundancy is usually present making it possible to construct such models. This subject area of *data compression.*

It is important to realise that when information is relayed through a network, the destination receives a *copy* of the source's message. The ability to *replicate* data presents another performance enhancement opportunity. Bit sequences of messages can be copied from the remote source to a secondary source which is local (closer) to the destination. *Mirroring* and *caching* are two methods of replication[1] which are commonplace within wide area networks like the Internet.

1.6.1 Compression

The information content of a message relates to the amount of "surprise" it invokes on the receiver and is function of the probability of its selection from a set of possible messages. So the information content I_j in the j^{th} message is given by

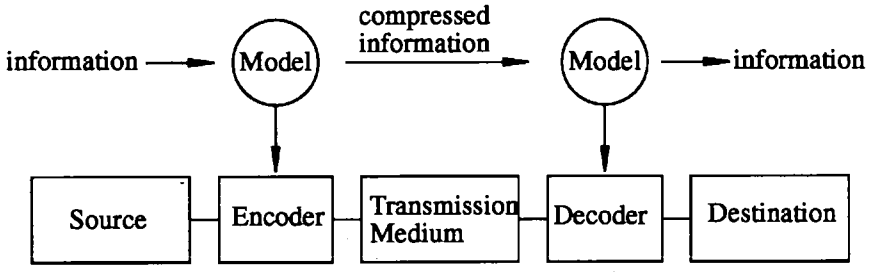
$$I_j = \log\left(\frac{1}{P_j}\right) \tag{3}$$

where P_j is the probability of selecting the j^{th} message. In a given session many (different) messages could be sent. Shannon's "A Mathematical Theory of Communication" [213] provides a means of deriving an average information measure for the source. Many encoding techniques (like ASCII) encode messages with equal numbers of bits despite the selection of messages having unequal probabilities. This introduces redundancy (reduces the surprise) so messages are longer than they need to be. This phenom-

1. Note that some sources [11] describe Mirroring as a replication technique [11] but not Caching. In this thesis replication is used as a generic term for Mirroring and Caching.

enon presents an opportunity to move some data (the predictable part) closer to the receiving user by employing compression techniques. In order to compress information something about its characteristics must be known. The knowledge is embedded in a model which is replicated at the source and destination. Prior to transmission the source passes a piece of information to an *encoder.* The encoder then uses the model to generate a compressed representation of the information which is then transmitted to the destination's *decoder.* The decoder then uses the (same) model to reconstruct the original message from compressed representation (see Figure 3).

Figure 3 A communication system with compression

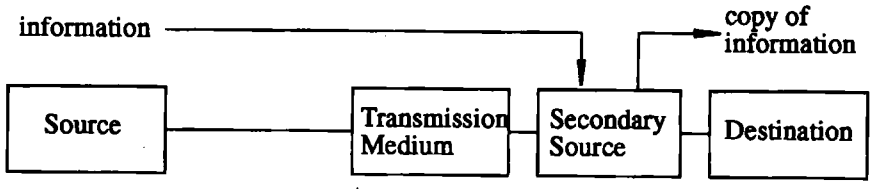


The encoder only sends the "surprising" part of the information. The "predictable" portion of the information can be generated from the model at the destination. In this way some of the data is brought closer to the user (at the destination).

1.6.2 Replication

Some messages are repeatedly transmitted across the network (for example popular Web pages). In "Information Theory" terms this represents a high redundancy (low surprise). Replicating such messages on a secondary source which is local to the destination can reduce traffic volumes on long haul transmission media Figure 4 shows a communication system with replication.

Figure 4 A communication system with replication



A replication technique used in the Internet is mirroring. It was reasoned that by duplicating popular sites at other geographical locations redundant document transfers over long haul transmission media could be avoided. Furthermore, in selecting documents from a (closer) mirror site users benefit from reduced access latencies. The `mirror` application works as follows:

- connect to the remote site,
- build the remote server's directory structure on the mirror,
- transfer the files and set their time stamp

The mirror connects to the remote server periodically and will duplicate any changes that have taken place. The process of duplicating changes generates network traffic which will impact user performance. Examination of network utilisation typically reveal peak periods of traffic intensity [234] as users only have "sporadic need for resources" [103]. Given that users have "temporal" access preferences the transfer of data due to changes on the remote server can be done at off-peak times rather than network busy periods.

Mirroring replicates entire sites whereas caching replicates on per document basis. Caches are not exclusive to wide area networks and are used in an number of memory hierarchies. Caches are used extensively in computer systems as a means of compensating for speed differences between various system components. For instance, in the 1960s [106] CPU speeds began to outstrip the access speeds of main memory (which

were increasing in size) which resulted in a bottleneck in the fetch-execute cycle. CPU performance was enhanced by placing a faster auxiliary memory (the cache) between the CPU and main memory. A definition of a cache is offered by Hill

> A cache is a small, fast buffer in which a system keeps those parts of the contents of a larger, slower memory [106].

World-Wide Web servers are merely an extension of the memory hierarchy and therefore subject to performance enhancements through caching. Caches were first implemented on client Web browsers using both main memory and backup disk [12].

Browser caches only operate on a per user basis, it was soon realised that further performance benefits could be brought about if the cache could be shared amongst many users [11]. This was done by implementing caches on *Proxy Servers.*

Proxy server were originally developed as a security feature whereby users from within a firewall (corporate Intranet) could access the public Internet World-Wide Web. Browsers operating from within a firewall do not make direct requests to remote servers, instead they send the requests to the proxy server which fetches the requested file on their behalf. The first request of a Web document is fetched from the Web server on which it resides and stored on local disk. Subsequent requests for the same document are served from the cache rather than original Web server.

1.7 Chapter Summary and Aims

It is thought that the impact on society of the "Communications Revolution" will be as significant as that of the Industrial Revolution [100], since advances in communication technology has resulted in many millions of networked computers throughout the world. The relay of information between source and destination requires the transmission of packets over a network. Low packet latencies facilitate rapid transmission of messages which is beneficial to any individual user's productivity. If the network per-

formance can be sustained the aggregate *social* productivity increases as the number of users increases.

Concerns about network traffic jams are such that the US government are planning to invest $500 million on building an Internet II "just so that the scientists can get some work done again" [205]. Companies also are building their own internets (commonly referred to as Intranets), mainly for reasons of security, but also as way of avoiding the "clutter and traffic" [205] that exists on the public Internet. Increasing resources (and accepting the incurred costs) is necessary for improving performance and it is likely that technological advances will continue to raise the capacity to transmit information.

Increasing the capacity of existing resources (such as bandwidth) is a common "solution" to wide area network performance issues. However bandwidth upgrades can be expensive and they do not address all wide area network performance problems. Benda [20] (amongst others) believes that bandwidth in the future will be "cheap and plentiful" such that the network will not present a bottleneck. However this is contrary to Morningstar's opinion that "no matter how much capacity is available, you always want more" [174]. The author tends to share the opinion of Morningstar rather than Benda and believes research into alternative methods of improving network performance is necessary.

Data compression techniques can be utilised when the volume of data threatens to overwhelm the communication facilities. Also, replication techniques, such as Mirroring and Caching bring data closer to the user thereby reducing propagation delays.

Thus the aims of this thesis are to analyse the delay in wide area networks and to investigate the techniques for reducing delay. A model of packet delays in wide area networks is constructed and experimental data is drawn upon explain the contribution that

each component of the model makes to the over delay. This delay model is then used to direct the research in to improving wide area network performance.

Game theory is used to analyse why congestion problems occur in wide area networks. Users are modelled as independent decision makers trying to get the most out of the system. This work was first presented by the author in [109]. MacKie-Mason et. al. [154] and Gupta et. al. [96] have since published work that uses a similar view point.

There are a number of opportunities to use compression within a wide area network environment. Messages can be compressed by the sender prior to transmission (to the receiver) over the network, or the packets in which the message is encapsulated can be compressed while in transit through the network. A number of experiments are carried out in order to determine whether compression and decompression is best done by the sender and receiver respectively or by the network.

The delay model is developed further in order to analyse the productivity benefits of mirror sites. The aim is to show that bringing data closer to the user reduces propagation delays and traffic on the long haul communication facilities. An experiment is carried out using a software development application which accesses data from a remote server in order to confirm (or refute) the results of mirror analysis.

Compression operates on the premise that data can be reduced if something known about it. It is argued in this thesis that replication techniques can also benefit from an understanding of how users access data. Mirrors operate on a very simple user access model based upon site popularity. However, caches use a more refined model (and are therefore deemed by the industry to more effective [175] than mirroring) whereby document accesses exhibit locality of reference. The effectiveness of caching with regard to bringing about improvements in wide area network performance is determined by the policies that are adopted for selecting what data to replicate.

It is when a cache reaches its capacity limit that it must put these policies in to effect. The log files of an Web proxy server are studied in order develop a model of cache occupancy growth. Experiments using trace simulations are carried out in order to investigate various caching policies. The conclusion of this research is that knowledge about how users access data is required to bring about improvements in wide area network performance.

CHAPTER 2 *Wide Area Network Performance*

This chapter presents a model of packet delay. The purpose of developing this model was to gain an understanding of wide area network performance and what can be done to enhance it. CHAPTER 1 explained that packet switching was developed in order to make efficient use of communication facilities. The speed of transmission of packets through a wide area network is affected by the bandwidth (data rate) of communication links, the processing power of routers, the signal propagation delay and time spent waiting in buffer queues due to traffic load.

Unlike circuit switched network, transactions in a packet switched network do not have access to dedicated resources. Instead resources are shared with a consequent likelihood of contention and delay in contention resolution.

There has been extensive research into the performance of shared networks (and other computing resources) using queuing theory. Users are viewed as jobs or tasks that arrive, wait and leave queues, but in this chapter users are presented as a set of interacting decision makers with conflicting interests. Game Theory is used to analyse the effects users have on each other's performance.

2.1 User Perceived Performance

Generally users prefer rapid response times to slow ones. Long delays can lead to more frequent errors [216] which results in user frustration, dissatisfaction with the system [216] and reduced productivity.

In using a computer system, human operators make errors which they need to correct. The penalties for error become high as delays increase. Consider the following simple example. In the Unix environment errored key-strokes can be deleted using the "erase" character [202]. The erase character is generated from the user's keyboard by typing the appropriately configured terminal key. Typing the erase character has the effect of "removing" the previously typed (errored) character from the command line. If the displayed effect of the erase character is delayed beyond the normal expectation of the user, the user may assume that the character has been lost in transmission and repeat the action. When (both) the erase characters eventually get through, two characters on the command line will be deleted (in some instances the delays can be such that the user deletes multiple characters). Understandably, frustration will result.

What users consider acceptable response time delays is dependant on three factors:

- Established expectations
- Individuals' tolerance to delays
- Users' adaptiveness to delays

Previous experience of the time to complete a given task influences users' expectations. For instance, PCs have become more wide-spread [182] and an increasing number of people have become exposed to them. A user of a PC is the sole user of the system so does not experience delays due to contention. Such a user may take these expectations to a multi-user system like a network environment.

Similar problems can occur with new systems. When a system is initially installed, the load is light and the performance is good. As the system becomes loaded (with more users) the performance degrades. As more users are introduced to the system the original users will perceive a deterioration in response below that of their expectations (established when it was first used it under low load conditions). Whereas users introduced to the system at a later date will perceive the performance as normal.

The following experiment was carried out in order show the correlation between performance and system load. The Dhrystone benchmark [38] was developed in 1984 by Reinhold Weiker and measures integer operation performance. The Dhrystones benchmark was executed throughout the day (using Version C/1) on a SUN 690MP server running Solaris 4.1.3, which at its peak, had 60 simultaneous users logged onto it.

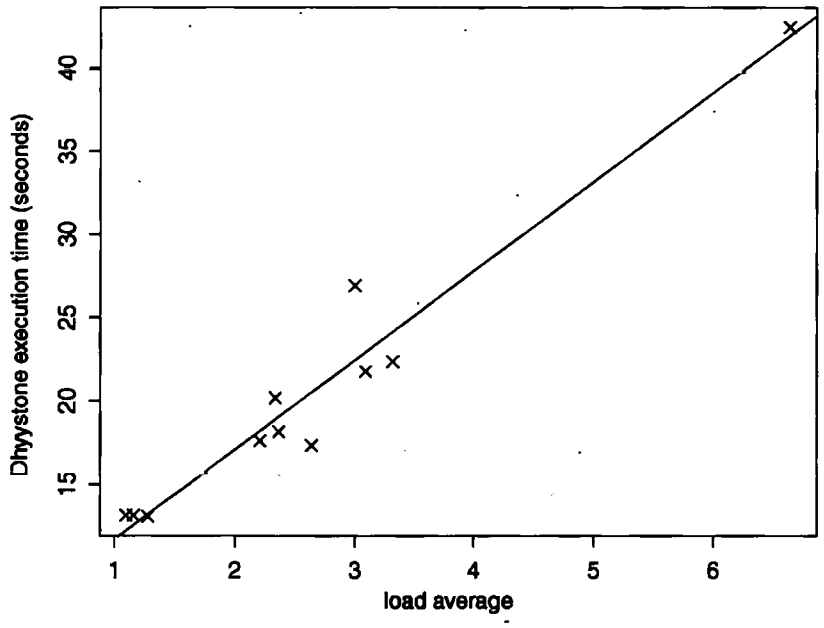
Figure 5 shows the Dhrystones execution time plotted against average system load (given by the `uptime` command). It can be seen that there is a deterioration in performance (increased execution times) as the load on the system increases.

Some users are more tolerant to delays than others. For instance, novice users may be more tolerant of longer delays than experienced ones [216]. Delay tolerance is also application dependent. File transfer users will not be as sensitive to a few delayed packets as users of virtual terminals. With a virtual terminal client application (operating in echoplexing mode) each key press results in a packet being sent to the remote server. This in turn results in a packet being returned from the server containing the "echoed" character. Each delayed packet will be directly experienced by the user of the virtual terminal application. A single file transfer request on the other hand results in many packets being sent from/to the server. If some of the packets in the request are delayed it is unlikely that the user will notice any significant effect in the overall transfer time of the file.

Response time expectations are also dependant upon how adaptive users are. If response times are slow, users have the ability to adapt their working practises. For instance, by reducing their system interaction frequency. If response times are sufficiently high, users "will fill in the long delays by performing other tasks, daydreaming, or planning ahead" [216]. Frequently however, when response times are high, users will complain.

Communication between two distributed computer systems results in an exchange of data (whether it be a single character of an entire file). Data is transferred over the network in packets. Each packet is subject to some delay in reaching its destination as a result of the transfer process. In wide area networks packet delays can be quite substantial and can contribute significantly to the overall system response. So much so that they are "rapidly becoming the largest single cost in distributed systems" [191].

Figure 5 Dhrystones performance (3rd March 1993)



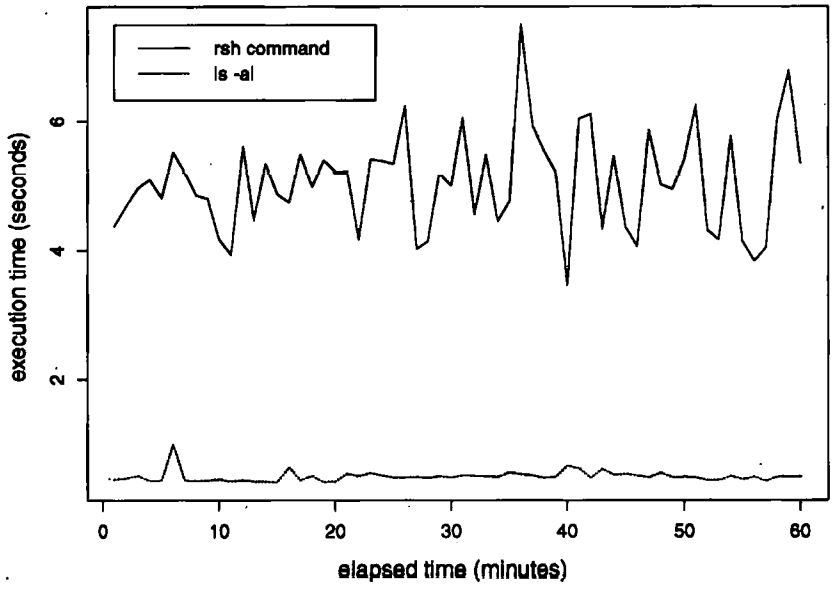
This is illustrated in by the following experiment. The command line below was executed on a local Unix workstation in the UK

```
time rsh ihgp2 "time ls -al /etc" > /dev/null
```

The effect of this command is to spawn a shell process on a remote machine in the US (IHGP2). From that (remote) shell process an `ls -al /etc` command was issued in order to generate a listing of the `/etc` directory on IHGP2. The `time` command was used (twice) to record both the time to execute the `ls -al` process on IHGP2 *and* the `rsh` process on the UK workstation.

The command line was executed at one minute intervals over a period of an hour during the UK's afternoon peak hours (and the US's morning peak hours). The results are shown in Figure 6.

Figure 6 Remote shell results



The mean execution time for the `rsh` command was 5.06 seconds, whereas the mean execution time for the `ls -al` command was only 0.50 seconds.

The time to complete the `ls -al` (on the remote IHGP2) process is purely the time to access the CPU and disk resources required to execute the command, while the time to

complete the `rsh` process is the time to execute the `ls -al` command plus the time to exchange packets of data between the local workstation and IHGP2. So the difference between `time rsh...` and `time ls -al...` is due to the overhead of the wide area network between the UK and the US. It can be seen that the network makes the most significant contribution to the delay and its variability.

Packet delays in a network are dependent upon a number of factors. In a multi-user environment one shares the resources with other users, so there is always the potential for contention and the consequent delay in resolving the contention. Effects on performance become apparent when the system is busy and users spend significant amounts of time waiting for other users' tasks to complete. In other words the factor that can most affect any single user's performance is other users.

Packet delays are particularly high when networks are busy. Indeed user perceived performance problems are typically a result of the mechanics that resolve contention [236], that is packets delayed in buffers waiting to be transmitted over some shared communications facility.

There are various ways of reducing packet delays but not necessarily without cost and very often systems designers and administrators find themselves caught between those who control the budget and users with ever growing appetites for more sophisticated applications [236].

High performance computers and the introduction of high resolution graphics capabilities have made computers more interactive and image oriented. Computing in the "visual domain" (possibly including some audio) is increasing in popularity. McCormick et al outline benefits of the visual domain

> Visualisation is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualisation offers a method for seeing the unseen [163].

Such functionality enables people to work more effectively and complete tasks in less time [68]. Users of computers systems tend to view performance in these terms and Shneiderman has some bad news for network administrators and designers

> Overall, the bandwidth of information presentation seems potentially higher in the visual domain than media reaching any of the other senses [216].

2.2 Packet Delays

Gray [90] presents the concept of computer systems as distributed systems which vary only in their degree of geographical distribution. Distributed systems pass messages in order cooperate and it has been shown above, where systems are distributed over a wide area, the time to pass messages can be significantly high and can impact the overall performance of the system.

The primary sources of delay in a packet switched network are introduced by the communication links and the routers. Figure 7 shows a diagram of a router connected to a communications facility. Partridge et. al. provide the following description of the architecture of a router

> [A] collection of network interfaces, some sort of bus or connection fabric connecting those interfaces, and some software or logic that determines how to route packets among those interfaces [192].

Gray [90] presents the following model of message delay, where a message can be a single byte, a single packet, multiple packets or a whole file

$$\text{Delay} = \text{Transmit\_Delay} + \frac{\text{Message\_Size}}{\text{Bandwidth}} + \text{CPU} \quad (4)$$

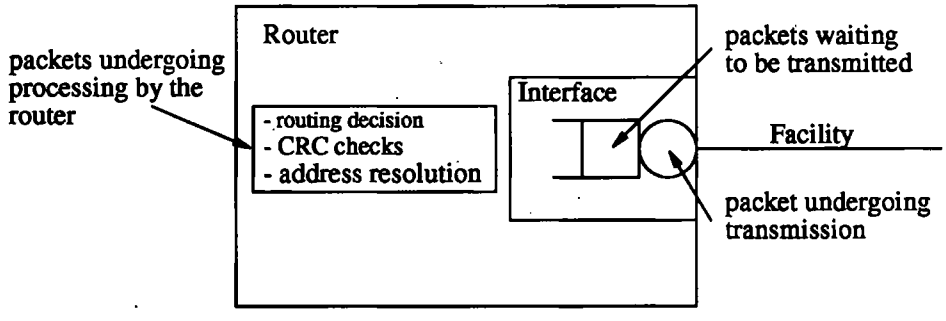
When a router receives a packet it examines the packet's destination address, makes a routing decision, calculates a CRC check, and does physical layer address resolution (it may also have to do compression). All of which results in some overhead in *processing*

the packet. This is represented by the CPU delays in Gray's model (4). Clearly processing delays accumulate with the number a network routers (bridges, switches, etc) a packet has to be "processed" by. The interface hardware for the communications facility is responsible for the actual transmission of packets.

The delay introduced by the router's transmission process is represented by the Message_Size/Bandwidth component in Gray's model. The interface will also have buffer memory to store packet waiting for transmission. Gray states in [90] that his interest "is in orders of magnitude" so delays waiting in buffer queues are ignored. However the incorporation of such delays are dealt with in this thesis and are described below.

Tranmit_Delay is the time it takes a signal to propagate through the transmission medium. It is independent of the routing device but can be significant in wide area networks.

Figure 7 Router configuration



The focus of this thesis is the transmission of packets of data, so Gray's model will be adapted to examine packet delays rather than message delays.

2.2.1 Processing Delay

Upon receiving a packet, a router has to perform an number of functions (described above) before it can pass it to one of its network interfaces for transmission. The time it

takes to carry out these functions is the processing delay D_C (equivalent to Gray's CPU component).

The packet routing engine can be represented as a complex systems of queues and processing units. In a local area network, where propagation delays are low and data rates are high, processing delays can be significant. In local area network environments and high bandwidth wide area networks such as ATM [201] there are pressures [192] to improve the efficiency of switching/routing fabrics [66] and develop lighter weight protocols [191].

However in current wide area networks where communication facility speeds are relatively low, router processing rarely represents a bottleneck. For the purposes of the analyses carried out in this thesis processing delay is frequently regarded as negligible.

2.2.2 Waiting Delay

Upon processing a packet, the router places it in a buffer associated with the appropriate network interface. Gray's model does not account for the time spent in this buffer awaiting transmission (something that Gray acknowledges in his paper). The purpose of Gray's model is to examine the effects of the technology rather than resource contention. However in this chapter particular emphasis has been put on effects on performance due to other users and so buffer delays have to be addressed.

An M/M/1 queue is one of the most commonly used queues [123]. It is used to model single queue systems where user interarrival times and service times are exponentially distributed. The expected delay D_Q for an M/M/1 queue [137] is given by

$$D_Q = \frac{1/\mu}{1-\rho} \tag{5}$$

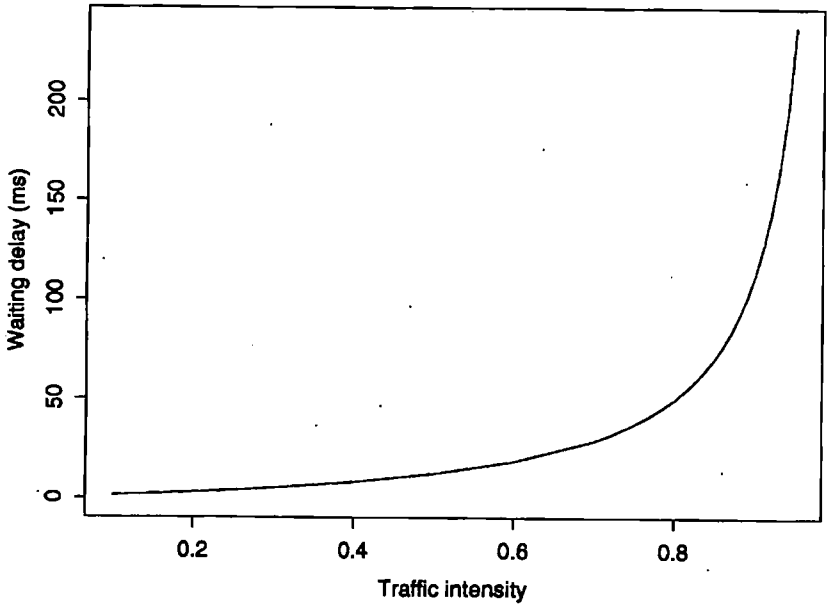
where μ is the bandwidth of the link, which is expressed in packets per second, and $0 \leq \rho < 1$ is the utilisation

If the link is not busy, that is $\rho \rightarrow 0$, then $D_Q \rightarrow 1/\mu$ which is the service time in queuing terminology. The mean time spent waiting in the buffer (not including time in service) D_W is given by

$$D_W = \rho \frac{1/\mu}{1-\rho} \tag{6}$$

The graph in Figure 8 shows the waiting delay for a 64kb/s link and mean packet size of 100 bytes. The graph illustrates the time spent waiting in buffers. It can be seen that delays increase steeply with load.

Figure 8 Waiting delays



2.2.3 Framing Delay

The framing delay is the time it takes to clock each binary digit (bit) of a packet (message) on to a communication facility and is a function of the channel bandwidth. The framing delay D_F is given by

$$D_F = \frac{\text{packet size}}{\text{bandwidth}} \quad (7)$$

(and is analogous to the Message_Size/Bandwidth component in Gray's model). It can be seen from (7) that the framing delay can be improved by either reducing the size of message or increasing the channel bandwidth (or both).

An experiment was carried out in order to determine the effects of message size and channel bandwidth on packet round trip delay. Round trip delays were measured for packet payload sizes (that is, the size of the packets excluding the protocol headers) of 64, 128, 256, 512, 1024 and 1472 bytes. These measurements were recorded for channel capacities of 64, 128, 256 and 512 kb/s over a network between Lucent sites in the UK and the US (refer to Appendix C for details on the experimental environment). This experiment was carried under conditions where the framing delays and the propagation delays were dominant over the processing speed of router which could therefore be ignored. Furthermore this experiment was carried out over a non-production network. That is the network was carrying no user traffic so the effects of waiting delays were negligible.

The effect of varying the channel capacity accounted for 28.99% of the variation in round trip delay, whereas the payload size of the packet accounted for 45.97% of the variation with 21.70% due to the interaction between the two factors (bandwidth and packet payload size). The remaining 3.35% must be attributed to experimental error (The packet round trip delay results are shown in Figure 103 Appendix C).

This experiment showed (that for the levels of factors used here) that packet size has a greater effect on performance. In which case it may be necessary to look to data compression technology to reduce packet sizes as a means of improving round trip delays.

2.2.4 Propagation Delay

The propagation delay is the time it takes a signal to travel the distance of a communications path (equivalent to Gray's Transmit_Delay). The speed at which the signal *propagates* is determined by the speed at which light travels through the medium of the communications channel. Table 1 shows examples of round trip propagation delays (Partridge [191]) for various terrestrial networks.

Table 1 Round trip propagation delays

Local Network (1.5 km)	17μs
Local Network (10 km)	110μs
Wide Area Network (5,000 km)	55,600 μs
Wide Area Network (14,000 km)	155,600 μs

Propagation delays are bounded by the speed of light, and while they are small for local area networks (due to the short distances) they can be quite significant for wide area networks.

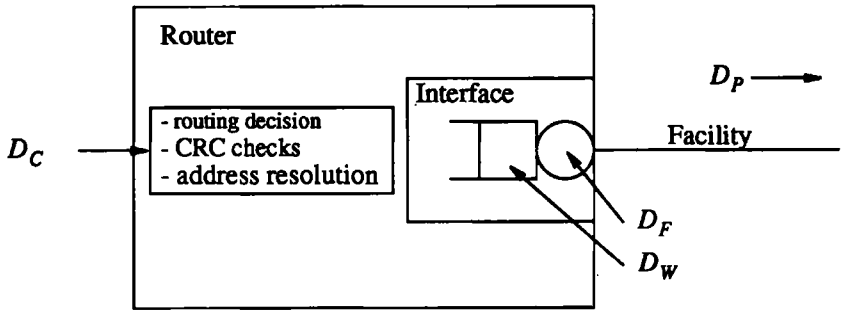
2.2.5 Delay Model

The delay model is revisited here. Given that D_L is the total packet delay (latency), then

$$D_L = D_C + D_W + D_F + D_P \quad (8)$$

where, D_C is the processing delay, D_W is the waiting delay, D_F is the framing delay and D_P is the propagation delay. The diagram in Figure 7 is repeated here in Figure 9 with the component delays of the model included.

Figure 9 Router and packet delays



The processing delay D_C is dependent upon the level of hardware resources (CPU and memory) in the router. The resources in a router are just like those of any other computer system in that they are subject to contention. The load on a router rises with the rate at which packets arrive at its interfaces. At Lucent technologies experience has shown that in a wide area network environment where communication facility bandwidths are low, the packet arrival rates are not sufficiently high to cause contention within the router that is significantly detrimental to performance. However in local area networks, where bandwidths are 10 or even 100 Mb/s, packet arrival rates can overwhelm a router. That is reducing D_F (increasing bandwidth) could impact D_C.

Forwarding packets between networks is not the only task performed by a router. It also has to compile routing tables from which to make routing decisions. As the Internet grows, the need to adopt more sophisticated routing protocols increases, which in turn places more demand on routers' processing resources.

D_P is dependent upon certain physical constraints, that is the speed of light. The framing delay D_F is dependant upon the state of current technology and economic constraints.

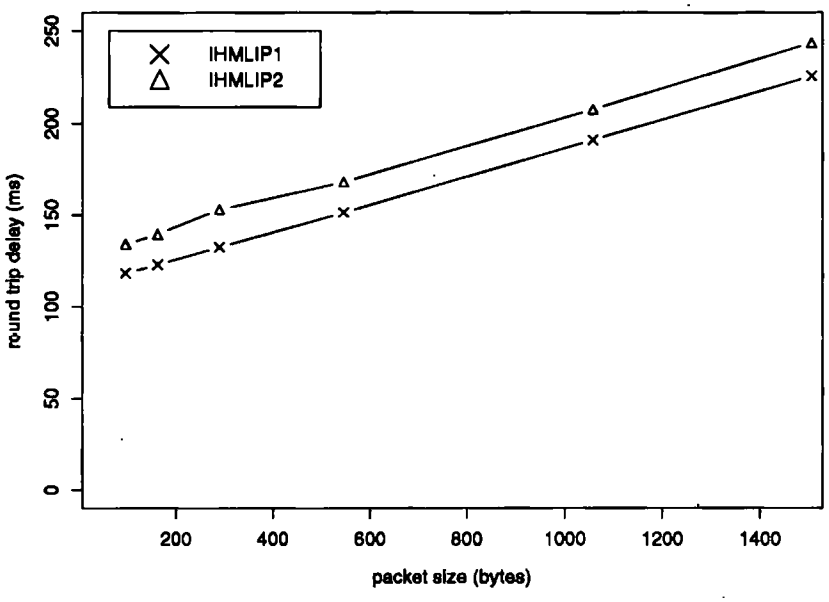
Appendix C describes experiments carried out to determine packet round trip delays across wide area networks. These experiments were carried out during network quiet

periods in order to counter the affects of load. Figure 103 shows the round trip delay results across a communications facility between the US and the UK for varying bandwidths. From these graphs, values for D_P and D_F can be determined. It should be noted that D_P and D_F are *one-way* latency delays, whereas the graphs represent the results for round-trip delays.

The slope of the graph in Figure 103 depends on the framing delay D_F, so that the slope of the graph flattens as the bandwidth increases and thus the contribution of the framing delay to the overall delay diminishes.

The intercept corresponds to the propagation delay D_P. The graph in Figure 10 shows two communications facilities of equal bandwidth. Both facilities connect to the same sites but are routed over different paths. It can be seen that the delay curves for both facilities are parallel to each other which shows that they operate at the same speed (224kb/s). However the points at which the delays curves would intercept the $y-$axis are different, corresponding to the difference in propagation delay between the two (diversely routed) facilities.

Figure 10 UK to US packet round trip delays



These results show the significance of propagation delays. The curves in graph of Figure 109 show round trip delays for facility speeds of 1.544 Mb/s. It can be seen that the slopes of the delay curves are very flat. As technology progresses and faster communications facilities become more economically viable, this slope will flatten further. However this will not in any way reduce propagation delay. Therefore distance and the speed of propagation will remain a major performance bottleneck for wide area networks.

Experiments in quantum optics have revealed that events can influence each other faster than a signal have could travelled between them. However, "nature has been clever enough to avoid any contradiction with the notion of causality" [44] such that *faster than light* communication is not (according to accepted theory) possible.

The waiting delay D_W is a function of D_F and ρ (the ratio of traffic volume to channel capacity). There is a desire (whether it be politically or market driven) to make network functions widely available, which means more users. Each user will contribute to the

volume of network traffic. While the amount of traffic will vary from user to user, ρ will increase as the user population increases.

High values of D_W impacts each individual user's productivity or working practises. Therefore there is a conflict between providing wide-spread access to network technology and acceptable levels of performance.

2.3 Game Theory

Game Theory [200] provides a means of analysing the rewards gained by interacting decision makers with conflicting interests. These conflict-of-interest scenarios are called *dilemmas*. Shared Packet switched networks introduce such dilemmas. Network performance becomes an important factor in determining the rewards gained from using it. In using communication resources (for example, bandwidth) an individual user denies others of those resources and therefore affects their performance.

Clearly a packet switched network can benefit many users, however as the number of users on the network increases, performance degrades for each user, that is, the individual rewards gained from using the network diminish. At some point the rewards will reach a level where they are equal to that of undertaking a task without using the network.

2.3.1 Prisoner's Dilemma

The Prisoner's Dilemma is a two player game ([173] p127) where each player adopts one of two strategies, either cooperate or defect. The "game" can be applied to many situations, but is commonly described in terms of two prisoners suspected of committing a crime (together). Each prisoner is interviewed separately and is given the opportunity of betraying the other by turning Queen's evidence. In betraying one's accomplice, a prisoner is adopting a defection strategy. However, a prisoner may elect to keep silent and therefore adopt a strategy of cooperation (in relation to the other prisoner not the

police). The sentence each prisoner receives depends not only on his own strategy, but also the strategy adopted by his accomplice. If both elect to cooperate (with each other) and keep silent, then the police can't make the main charge stick and both get a shorter prison sentence for a lesser crime. On the other hand if both choose to defect and implicate the other prisoner, then both are found guilty of the main charge and therefore receive a longer sentence.

At this point both prisoners are better off keeping silent as it ensures a shorter sentence. However, the dilemma arises when the prisoners adopt different strategies. A prisoner can bargain for his freedom by implicating his silent accomplice. The accomplice then takes full responsibility for the crime thus receiving a longer sentence than if he himself had confessed.

In general, if both players cooperate they both receive a payoff $\pi_{CC} = R$[1] (Reward for mutual cooperation), while two defecting players get $\pi_{DD} = P$ (Punishment for mutual defection). A player unilaterally defecting will receive a payoff of $\pi_{DC} = T$ (Temptation payoff) while the cooperating opponent receives $\pi_{CD} = S$ (Sucker payoff).

For a Prisoner's Dilemma to exist, $\pi_{DC} > \pi_{CC} > \pi_{DD} > \pi_{CD}$ must be true[2]. Table 2 shows an example payoff matrix for the Prisoner's Dilemma. The value to the left of the comma is the payoff (sentence in years) for the row player's strategy, the value on the right is the column player's payoff.

Note that the players are not playing against each other, that is, each "prisoner" is not trying to get a lighter sentence than his accomplice, he is just trying to get the best payoff for himself (the lightest sentence). In order to be set free he needs to exploit his accomplice's good will in keeping silent. Reciprocating the gesture will result in a one

1. Where C = cooperate and D = defect.
2. Note that the "greater" payoff with the Prisoners Dilemma is the sentence with the least number of years

year prison sentence. Furthermore, in cooperating, a prisoner risks being exploited himself if his accomplice confesses.

Table 2 Prisoner's Dilemma payoff matrix $\pi_{DC} > \pi_{CC} > \pi_{DD} > \pi_{CD}$

	Cooperate	Defect
Cooperate	1, 1	3, 0
Defect	0, 3	2, 2

The "solution" to this problem is always to defect as, no matter what the other player does, defecting always ensures the best result. At worst, the rational prisoner will get a two year sentence (if his accomplice is also rational). There is a chance of being set free if an irrational accomplice shows him any loyalty. In the case of the Prisoner's Dilemma the defection strategy is called the *equilibrium* point. An equilibrium point (or points) exists when neither player can benefit from a unilateral change in strategy.

2.3.2 Chicken's Dilemma

The Prisoner's Dilemma is applicable to many social situations. Frequently, the actions of the members of a community affect each other. In a shared network environment, each user consumes some of available bandwidth leaving less for others. A user may elect to limit their use of the network in order to be fair to others or they may just consume bandwidth as necessary. Unfortunately the performance experienced by any user is dependent upon the strategy adopted by others (who may or may not regulate their usage).

Suppose that two users need to transfer large files across a network. The file transfer places a load on the network which is 60% of its total capacity. If both users initiate their file transfers simultaneously (and we assume that they will), the offered load on the network will be 120%! The network will have problems coping with an offered load

which is greater than its maximum capacity (100%), typically bad performance will result. Each user has access to compression software which achieves a 2:1 compression ratio, and if they use it the file transfer only places a 30% load on the network. If both users compress their data (cooperate), the total load will only be 60% and performance will be just fine. However, if one user unilaterally compresses his data (cooperates while the other defects), then the total load is 30%+60%=90%. The data throughput over the network will be worse than if both users had compressed their data, but the network should be able to cope. While the cooperating user has less data to send (because it has been compressed) his packets will get delayed by those of the defecting user (who has more to send). Furthermore the defecting user does not incur the delays of the compression process. The payoff matrix for this dilemma is shown in Table 3.

Table 3 Payoff matrix for transferring compressed files (Chicken's Dilemma)

	User B compresses file (cooperate)	User B does not compress file (defect)
User A compresses files (cooperate)	2, 2	1, 3
User A does not compress file (defect)	3, 1	0.1, 0.1

For the Prisoner's Dilemma, the payoff ordering is $\pi_{DC} > \pi_{CC} > \pi_{DD} > \pi_{CD}$. The dilemma shown here is known as the Chicken's Dilemma. The payoff ordering is $\pi_{DC} > \pi_{CC} > \pi_{CD} > \pi_{DD}$, which for the example in Table 3 is demonstrated by the inequalities 3>2>1>0.1. Close examination of Table 3 shows that the game of Chicken is where two "players" drive towards at each other at speed, the first one to swerve is the loser and referred to as "chicken". Normally the payoff π_{DD} is 0 (or negative), because both drivers end up dead. The π_{DD} payoff for the "Compression Dilemma" is positive,

but nevertheless small (very slow transfer rates). With the Prisoner's Dilemma the rational strategy is to defect, therefore if both players are rational the outcome is DD (the equilibrium point). With the Chicken's Dilemma the DD outcome results in the worst payoff for both players. If one player changes his strategy to cooperate, he increases his payoff ten fold (given the payoff matrix in Table 3). However, the benefit to the defecting player is even greater. Even though the cooperating player is being exploited by the defector, cooperation yields a higher payoff. Remember, with such dilemma games, players do not "compete" against each other, instead they play against a "third party", the objective being to maximise one's own payoff. The Chicken's Dilemma has two equilibrium points, CD and DC, meaning, a unilaterally cooperating player will not increase his payoff by adopting a defection strategy.

2.3.3 Iterated Prisoner's Dilemma

The Prisoner's and Chicken's Dilemmas described above are "one shot" games. With the Prisoner's Dilemma, defection is the rational strategy. With the Chicken's Dilemma, a player can avoid the worst case payoff by cooperating even if the other player does not, in short, neither game encourages mutual cooperation. Yet mutual cooperation does occur in the real world. With a one-shot dilemma there is no opportunity to retaliate against a unilaterally defecting player. A variant of the (one-shot) Prisoner's Dilemma is the Iterated Prisoner's Dilemma, where players interact repeatedly. Iterated Prisoner's Dilemma encourages mutual cooperation because each player has the ability to punish the other for defecting, only a complete sucker would continue to cooperate against a constantly defecting opponent.

An iterated Chicken's Dilemma is less intuitive, particularly when it results in two fatalities (in the case of two drivers speeding towards each other). For two network users, the mortality rate is somewhat lower, and they may want to send multiple files. While uni-

lateral cooperation avoids the worst payoff, a player may choose to defect in order to encourage an opponent to cooperate.

2.3.4 N-person Games

Network communities usually consist of many users, not just two. Dilemmas that involve an arbitrary number of players are called N-person games. The N-person Chicken's Dilemma is called the Volunteer's Dilemma. This is where many people benefit from one person "volunteering" to undertake some task. Usually, the volunteer also benefits and is better off than if no one had volunteered (but would have been much better off if someone else had volunteered).

However, there are some Volunteer Dilemmas where it needs a certain number of cooperators in order to achieve a good payoff. Glance and Huberman describe how increasing environmental awareness has brought about voluntary recycling campaigns amongst social groups:

> Recycling poses a social dilemma for the consumer: the environmental benefits are great if most of the population recycles but marginal if only a few do [85].

Table 4 shows the payoff matrix for recycling campaigns. Although it is an N-person "game" it is reduced to a two person-like payoff matrix. The payoffs in the matrix indicate the payoff to individual members of the group that is cooperating or defecting.

Table 4 Environmentalist's payoff matrix

	Many people recycle (cooperate)	Many people don't recycle (defect)
The remaining few people recycle (cooperate)	2, 2	0.001, 0.01
The remaining few people don't recycle (defect)	2.1, 1.8	0, 0

Clearly, if no one recycles (DD) then there is no benefit to the environment, and therefore there is no benefit to beings living within it (in fact it may be detrimental, in which case the payoff is negative). If everyone recycles (CC), then there is benefit all round. If only a few people recycle (CD) they go to a lot of trouble for almost nothing, any benefits (albeit small), are also enjoyed by those that don't recycle (without any effort on their part). The DC payoff is almost the same as CC but does highlight the "Free Riders" phenomena in that a few people benefit from the environmental effects of wide spread recycling even though they do not contribute to the programme themselves. Complete mutual cooperation is rarely found within social groups, there are always some "free riders" (fare dodging, tax evasion, insurance fraud). Within a predominately cooperating community, groups of defectors can coexist. Such members of the community receive a higher payoff than the majority (for example, use of public transport without incurring the cost of the fare). Provided the number of defectors is kept relatively

small, the payoff for a cooperating majority is not significantly affected. Jamieson presents the defectors point of view

> We can reason: since my contribution is small, outcomes are likely to be determined by the behaviour of others [124].

Cooperators on the other hand consider the consequences of mutual defection, that is, "what if everyone did it?" (the Kantian point of view [110]). A shared network environment however, may not be able to tolerate too many free riders. Some network applications, such as video, have large appetites for bandwidth and a single user (with a high end PC or workstation) can place considerable load on a local area network. In a wide area network environment the problem is even more acute. Table 5 shows the payoffs for an N-person Compression Dilemma where cooperators compress their data and defectors do not. This is based on the Environmentalists Dilemma, except the "many-people defect" strategy is replaced with a "not-enough-people cooperate" strategy.

Table 5 N-Person compression dilemma payoff matrix

	Many users compress data (cooperate)	Not enough users compress data (defect)
The remaining few users compress data (cooperate)	2, 2	0.001, 0.01
The remaining few users don't compress (defect)	2.1, 1.8	0, 0

The load on the Internet and the payoff in terms of network performance experienced by each individual user can be modelled using a Volunteer's Dilemma which was illustrated by an experiment carried out by Science 84 magazine (described by Poundstone [200]). Readers were invited to send in requests for either $20 or $100. Each reader

would get what they asked for provided no more than 20% asked for the $100 payoff. If more than 20% asked for $100 then nobody would get anything. The defection rate was 35% so there was no payout. Users may elect to limit their use of the Internet (say, by using compression techniques or rescheduling their on-line activities to quiet periods) so that bandwidth is available for others, or they may use it regardless of what demands they place on the network. With the Enironmentalist Dilemma, the π_{CD} payoff (second to worst) occurs when the majority defect. When using bandwidth intensive applications over a network like the Internet, the π_{CD} payoff occurs even when the defectors are in the minority, as Poundstone states

> ...if too many people are greedy, no one gets anything [200].

Players in any dilemma are free to choose their strategy (cooperate or defect), and users of a network always have ability to "step on" one another. As Stoll states In the Cuckoo's Egg

> There's plenty of room for truly 'creative anarchy' on the networks as they are - nobody is in charge of them, nobody makes the rules - they exist purely out of cooperative effort and they evolve freely at the whim of their users [228].

However, as illustrated by the Science 84 magazine experiment this "cooperative effort" can be difficult to bring about when "players" are spatially diverse. This may be the case with wide area networks. In trying to maximise their own individual gains users can cause network congestion. The result is a degradation of performance not only for others but also themselves.

2.4 Chapter Summary

The delay model presented in this chapter illustrates where packets spend time during transit between source and destination. The analysis of the model draws upon experi-

mental results described in Appendix C to show how much each component delay contributes to the overall delay and what factors affect them.

Technological advances in telecommunications will make it possible to (continue to) reduce packet framing delays. This in turn will reduce the packet waiting delays due to contention resolution at communication facility interfaces. However there are other factors, both social and technical that affect interface queue sizes (and therefore waiting times). In a free and open network environment like the Internet the volume of network traffic is (almost) only limited by the users' desire to fulfil their productive requirements. Such forms of *congestion control* are very often unsuccessful, and on the Internet this has lead to communication resources (bandwidth for example) being over utilised [154]. Game theory was used to model the user "payoffs" gained from using shared network resources. It shows that "overgrazing" the can be counterproductive.

It is likely that technological advances in computer hardware and software will more than keep pace with advances in telecommunications technology, which helps users fulfil their desire to be (more) productive. While, framing delays, waiting delays, and router processing delays can be reduced by investment resources and technological solutions, propagation delays are unaffected. Propagation delays present a bottleneck which is currently beyond any foreseeable technological solution. According to Rheingold

> Until the speed of light barrier is broken, the physical size of the planet precludes a truly instantaneous on-line cyberspace; the larger your cyberspace is distributed geographically, the larger your system time lag is likely to be [206].

This chapter introduced methods whereby data can be "moved" closer to the user in order to overcome high delays due to signal propagation. Performance improvements can be brought about by replicating (mirroring or caching) data from remote sources at

a location which is nearer to the user. By fetching the (replicated) data from a nearer source than the original (remote) source lower propagation delays are experienced. Fetching data from a mirror or cache can also help to alleviate traffic congestion on busy facilities.

Compression is also presented as a method of bringing data closer to the user. A model of the expected data is shared by both source and destination. Therefore in order to convey a message to the receiver, the source only has to send the model parameters rather than entire message. This results in reduced packet sizes (or fewer packets), thereby reducing delays due to framing (less bits to send) and shorter queue waiting times.

CHAPTER 3 *Compression*

This chapter examines data compression as a means of improving wide area network performance. Compression is concerned with reducing the size of messages. Compression techniques can be used to reduce data storage requirements and transmission times through networks. Compression affects three components of the delay model (6) in CHAPTER 2: framing delay D_F, waiting delay D_W and processing delay D_C.

Framing delays are reduced because with compressed messages there are fewer bits to send. A consequence of reduced framing delays is that queue sizes are reduced therefore packets experience shorter waiting times. However compression requires processing resources which can lead to increases in D_C.

Research into compression has been extensive. However this research tends to concentrate on the development of compression techniques and the theoretical background of the subject. This thesis does not cover these areas. Rather, this thesis examines *where* compression should be deployed.

Compression tools exist on the end user machine like the Unix Compress command for reducing files in order to optimise disk storage. Network messages are often stored on hosts as files so such compression tools could be used to reduce message (file) sizes

prior to transmission. Also, some encoding techniques include compression as part of the technique, in particular graphical image formats (for example GIF and JPEG). The integrated circuit manufacturer Intel describes the benefits of doing compression at the *application level*

> The emergence or real-time compression software, along with the increase is system performance, has enabled the tradeoff of low-cost MIPS for limited bandwidth, and cost-effectively extended the life of existing networks [120].

An alternative to doing application level compression is to compress within the network. Intermediate nodes (router for example) in the network could compress packets prior to transmitting across a communications facility (of course the remote intermediate node has to decompresses it). Cisco, the router manufacturer advocates doing compression within the network [46].

With application compression, an entire message (for example a file or document) is compressed prior to transmission by the sender and then decompressed by the receiver. The result of application compression is that less packets are required to send the message.

With network compression, packets (from a message) are compressed in transit, typically at the output interface to a communications facility. Packets are decompressed at the input interface. Network compression does not reduce the number of packets but it does reduce their size.

It should be understood that compression works by removing redundant data from message. Once this redundancy is removed, if the compression method is an efficient one, a message cannot be compressed further (because there is no more redundancy to remove), therefore subjecting a message to multiple compression processes is of little benefit. It could even be detrimental to performance. There is therefore a choice to be

made as to whether to do compression at the application level or in network - or whether to do it at all. This chapter begins with a discussion of compression and the trade-off between the reduction in delay brought about by the transmission of less data and the processing time required to compress it.

In section 3.3 an analysis was carried out on a number of compression applications in order to assess their ability to compress data. This analysis used the Calgary Text Compression Corpus which is common set of data files used to evaluate compression methods. In addition an attempt is made to find a figure of merit that would provide a means of assessing the compression susceptibility of an item of data. This analysis is used later on in the chapter to examine the effects of protocol overhead on compression ratios.

Experiments were carried out in section 3.4 to examine the performance enhancements gained from doing compression at the application level compared to doing it in the network. A model was constructed of the data rates achieved by each compression method over various link speeds so that their performances could be compared.

3.1 Compression Preliminaries

Wide area network communications facilities "can consume a significantly large portion of the operating cost of the network" [46] and in most organisations (commercial, education or government) the economical constraints mean that wide area network facilities should be used in the most efficient way possible.

Compression techniques are not limited to electronic communication methods. In natural language there is an inverse relationship between word length and relative frequency. Zipf, in the Psycho-Biology of Language [248] examines the cause of this relationship. That is, whether the length of a word affects the frequency of its usage or whether the frequency affects the length. Zipf argues that word selection is based upon the meaning a speaker or writer wishes to convey rather than word length

> there seems no cogent reason for believing that the small magnitude of a word is the cause of its high frequency or usage [248].

Zipf therefore concludes that word lengths are reduced as their usage increases within a particular speech group.

One of earliest uses of compression in electronic communication was Morse Code where the most frequently occurring letters are assigned shorter codes than those that occur less frequently. As a consequence telegraphers were able send messages faster than if each letter was represented by a codes of equal length. Claiborne [52] relates an example from the early days in telegraphy where codes were used to represent whole messages which were commonly used. For example the code: MX01 could stand for "Happy Birthday".

According to Bell et. al. "data compression is inextricably bound up with *prediction*" [19]. Computer systems (such as the World-Wide Web) process many different type of media, for example text, images, audio and video. Any predictions regarding a data sequence will be influenced by the types of media it represents. Because of this there are many different compression techniques.

Text compression tends to fall into two categories: *statistical* and *dictionary*. Text data is typically encoded using fixed length codes (for example ASCII). Unless each text symbol occurs with equal probability any message encoded in this way will possess a certain amount of redundancy. Statistical compression works by assigning shorter codes to the most frequently occurring symbols (and longer codes to the least frequently occurring symbols). In ASCII a character would be 8 bits in length irrespective of its frequency of occurrence. Given that a particular character's probability of occurrence is p_c then ideally its encoded length should be about $-\log_2 p_c$ bits [212].

Dictionary compression uses the knowledge that in English, words and even word sequences are often repeated. Compression is brought about by "replacing a repeated string by a reference to an earlier occurrence". This technique was developed by Jacob Ziv and Abraham Lempel and is known as Lempel-Ziv or LZW compression.

Compression is inherent within many image encoding techniques (for example GIF and JPEG). There are two variants of image compression techniques:

- Lossless
- Lossy

With lossless compression techniques the recovered image after it has been compressed and decompressed, is exactly the same as the original. Full colour images are usually encoded (uncompressed) in 24 bits/pixel. According to Lane [140] lossless image compression techniques can achieve between 2:1 and 5:1 compression ratios.

JPEG uses a *lossy* compression technique, that is the resultant decompressed image is not exactly the same as the original. The human eye possesses certain limitations, for instance "small color changes are perceived less accurately than small changes in brightness" [140]. JPEG takes advantage of this knowledge and compression ratios of between 10:1 and 20:1 [140] can be achieved without any *noticable* loss in image quality. Furthermore the degree of lossiness can be adjusted such that compression ratios of between 30:1 and 50:1 [140] can be achieved. However the degradation of image quality becomes more apparent.

Different techniques tend to favour different image types. JPEG is better for "realistic" full colour or grey scale images such as those from scanned photographs. However GIF is better for images with small numbers of distinct colours such as line drawings [140].

Audio data is another media that utilises lossy compression. One common compression method used for speech is *linear prediction*. Encoder and decoder attempt to predict the

next value for a signal from a set of previous samples. Compression is achieved if the data sent to predict the next value is less than that of sending the actual value. One has to accept that the predicted value will not be equal to the actual value but provided it is close such errors can be tolerated. Besides quantization errors are inherent with digitised audio signals so the reproduction is always different from the original signal.

Video media also uses lossy compression (such as MPEG). Like still images some aspects of video images are deemed more important to picture fidelity than others and it is acceptable for received video images to be imperfect copies of the original. Video compression techniques also take advantage of the fact that images change very little from frame to frame. The amount of data transmitted can be reduced by sending only the changes between consecutive frames rather than the entire frame. Furthermore for certain applications such as general entertainment, some frame loss due to transmission errors can be tolerated without too much perceptible effect upon picture quality. Compression ratios of 26:1 [191] can be achieved using such techniques.

Many documents produced using word processors can be found on the World Wide Web. These documents can be quite large so authors sometimes compress them. While this can reduce access latency and network load, users who do not have access to the necessary compression tools are unable to read them.

One solution to this is to make both compressed and uncompressed versions of the document available (sometimes authors even make multiple compression versions available). In an ideal world only those users who do not have access to the necessary (de)compression tools will access the uncompressed versions, while those that do, will access the compressed versions.

However, those with access to compression tools have a choice of either format. Presumably, those concerned with reducing their access latency will choose to download

compressed versions of the documents. However those who do have such concerns may elect to download the uncompressed version despite the benefits of reduced network load.

In making uncompressed versions of a document available for users who do not have ability to decompress them, there is no way of controlling which users can access which version. By only making compressed versions available authors limit their readership to those users with access to, and familiarity in using the necessary compression tools.

Many Web pages contain inline images. Images files are quite large and constitute a high proportion of the data transferred as a result of a request for a page. Netscape Navigator gives the user the option to turn off automatic download of inline images. The result is data compression with some losses (no images). Usually such losses are only acceptable to users connected to the Web via modem lines where the data rates are typically low. In such an enviroment the benefits of this lossy compression technique are significantly reduced access times and thus smaller phone bills.

Another option provided by Netscape is to view images as they are downloaded (as opposed to viewing them only after transfer of the image is complete). This gives the user the ability to interrupt transfer of the image, once again resulting in lossey compression (in this case the loss is an entire section of the image).

Interruption of image file transfers are usually a result of a user requesting another page before the entire transfer is complete rather than intentional desire to compress data. In cases where inline images are only included for aesthetic reasons, the loss of the images does not impact the informational content of the page, that is the information is contained within the text. With browsers like Netscape Navigator, the text is downloaded and displayed first, and the inlined images after. The user can extract the informational content of the page (read it) in parallel with the download process of the images. There-

fore the image download process does not delay the user from reading the text. An individual user will benefit from reduced access delays by compressing their data (although they may experience some losses), however, in this case, the user is not significantly affected by the delays in downloading the inlined images. Therefore any individual user's desire to compress their data by interrupting the image download process would be a result of altruistic behaviour, as any benefits (increased bandwidth availability) would only be experienced by other users.

If the browser itself only downloaded the low resolution image and required a specific action from the user in order get the full resolution version (that is decompress it) then a better compression ratio may be achieved. Obviously this requires the developers of Web browsers to implement them this way. It also requires Web service providers to make more interlaced images available.

3.2 The Trade-off of High Compression Ratios

Clearly the higher the compression ratio the faster information can be transmitted across a network. However Partridge [191] warns that as the degree of compression increases encoding/decoding delays become unacceptably high. As Danskin states

> When the network transmits data as fast as the computer can produce it, spending cycles on compression is counterproductive. When the computer must wait for the network, cycles spent on compression are well-spent [63].

Equation (7) in chapter 2 shows that delay increases with message size. In order for compression to be beneficial the following must be true

$$\frac{\text{message size}}{\text{channel bandwidth}} > \text{compression delay} + \frac{\text{compressed message size}}{\text{channel bandwidth}} \qquad (9)$$

Consider the following example. NASA's Galileo mission to Jupiter suffered a set back when the high gain antenna failed to unfold fully rendering it inoperable. Galileo's only means of communication was via a low gain antenna which operated at a much lower transmission capacity compared to that of the high gain antenna. This meant the probe would be sending "far fewer data bits" [179] back to Earth.

As a result NASA adopted a "new telecommunications strategy" in order to increase the amount of information the low gain antenna could relay back to Earth. Part of the new telecommunications strategy involved reloading Galileo's on board computer with advanced compression software, which increased "the information content in each bit by a factor of 10 or more" [126]. One inevitable question that appeared in an FAQ[1] on one of NASA's Web pages was "Why weren't the new data compression methods originally planned to be used with the High Gain Antenna to get even more data back?". The response from NASA was as follows: "Galileo's computers are not fast enough to compress data at the rate which was to be transmitted through the HGA!"

Compression/decompression requires systems resources (CPU and memory) and will add a finite delay to the transmission process. It is necessary to determine if delays incurred by the compression process outweighs the reduction in delay gained from sending less data across the network.

Computer video games are typically at the cutting edge of computer processing speeds in their constant attempt to achieve both visual and audible realism. They are also highly interactive and realism can only be taken so far as the processing speed of the computer system will allow. PC games usually have a "minimum" and "recommended" system configuration. Minimum means that the game will not work on a system that does not meet this specification. A system with a specification higher than the minimum

1. Frequently Asked Questions

but lower than the recommended will be able to run the game, but the interactive performance may be slow.

Some games allow the user to reduce the visual sophistication of the game if the computer platform is not up to the recommended specification. This function is described in the instruction manual of the game Doom II

> The default setting for the screen detail is HIGH. If you have a slower computer or video card, and the action is too jerky, you may wish to select LOW to make the game action smoother [119].

The amount of data that has to be processed is reduced thereby improving the game's response time. However some of the realism (lower resolution of the screen's image) is traded (lost) for an improvement in performance (otherwise one has to buy a faster computer). These losses have to weighed against the performance improvements.

3.3 Analysis

There are typically three tools available on Unix systems for compressing files; `Pack`, `compress` and GNU ZIP. `Pack` and `compress` are two Unix system supplied commands which use Huffman coding and Lempel-Ziv coding respectively. GNU ZIP command `gzip` is a shareware compression tool that also uses Lempel-Ziv coding. Throughout this chapter these three compression tools will be referred to as, Pack, Unix and Zip.

These compression techniques are examined and the claims in the system documentation regarding their *performance* are investigated. Compression performance is typically expressed in terms of a compression ratio:

$$\text{compression ratio} = \frac{\text{size of original file}}{\text{size of file after compression}} \quad (10)$$

though compression performance can be expressed in other ways. For instance `Pack` states its compression performance in terms of how much data is "reduced to"

$$\text{percentage relative file size} = 100\% \times \frac{\text{size of file after compression}}{\text{size of original file}} \quad (11)$$

Whereas the Unix expresses compression performance in terms of how much data is "reduced by"

$$\text{data reduced by} = 100\% \times \left(1 - \frac{\text{size of file after compression}}{\text{size of original file}}\right) \quad (12)$$

as does ZIP. Both the "reduce to" and the "reduced by" metrics are quoted in percentage terms.

The Unix states a "reduced by" metric of 50 to 60% for text files and claims to be better than Pack. Indeed Pack claims that text data is "reduced to" 60 to 75%. If these metrics are converted to compression ratios then we get 2.0 to 2.5 for Unix and 1.3 to 1.7 for Pack.

ZIP is also based on Lempel-Ziv coding but claims an improved text "reduced by" metric of 60 to 70% [87] over Unix `compress`. This translates to a 2.5 to 3.3 compression ratio.

The de facto standard for evaluating data compression schemes is the Calgary Text Compression Corpus[1]. The Calgary Text Compression Corpus comprises 18 files containing a variety of data formats including; English fiction and nonfiction ASCII text; high level computer program code; executable code; geophysical binary data, and bit-

1. The Calgary Text Compression Corpus files can be downloaded from ftp://ftp.cpsc.ucalgary.ca/pub/projects/text/compression.corpus/text.compress.tar.Z

mapped pictures (see Table 6.) These files are used to compare the three compression tools described above.

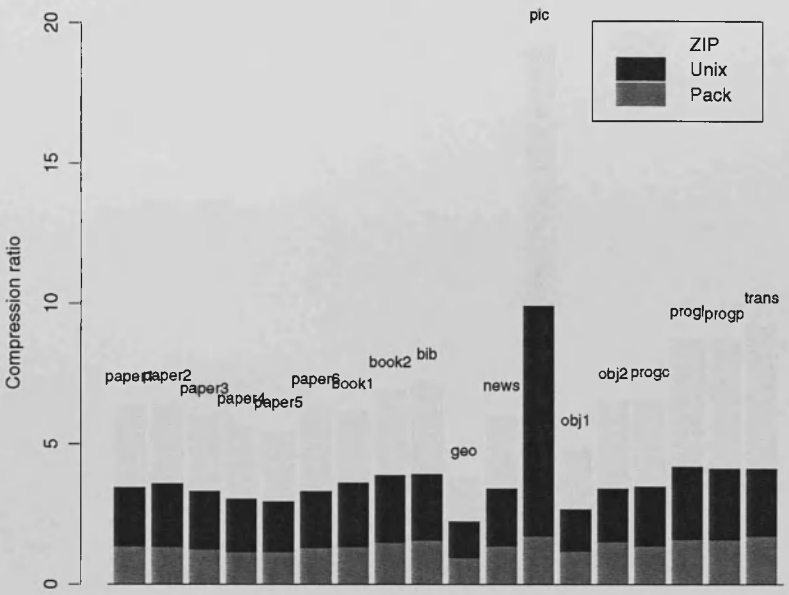
Table 6. Calgary Text Compression Corpus

	Name	Description
1	paper1	Witten, Neal and Cleary: Arithmetic coding for data compression
2	paper2	Witten: Computer (in)security
3	paper3	Witten: In search of "autonomy"
4	paper4	Cleary: Programming by example revisited
5	paper5	Cleary: A logical implementation of arithmetic
6	paper6	Cleary: Compact hash tables using bidirectional linear probing
7	book1	Hardy: Far from the madding crowd
8	book2	Witten: Principles of computer speech
9	bib	Bibliographic files (refer format)
10	geo	Geophysical data
11	news	News batch file
12	pic	Picture number 5 from the CCITT Facsimile test files (text + drawings)
13	obj1	Compiled code for Vax: compilation of progp
14	obj2	Compiled code for Apple Macintosh: Knowledge support system
15	progc	C source code: compress version 4.0
16	progl	Lisp source code: system software
17	progp	Pascal source code: prediction by partial matching evaluation program
18	trans	Transcript of a session on a terminal

The compression ratios for each file are shown in the bar chart in Figure 11. The length of each shaded region corresponds to the magnitude of the compression ratio.

On average Unix achieved a compression ratio of 2.5. ZIP performed somewhat better with an average compression ratio of 3.1 and Pack came out worse with an average compression ratio of 1.8. These results are consistent with the claims stated above.

Figure 11 Pack, Unix and ZIP compression ratios



For any unit of data the compression ratio will vary depending upon its susceptibility to the method used.

It can be seen from Figure 11 that some files compress better than others. It would be useful to know some property which relates to a file's susceptibility to compression. In Figure 12, Figure 13 and Figure 14, file size is plotted against compression ratio for the three compression tools. It can be seen that there is no strong linear correlation between file size and compression ratio. Indeed the correlation coefficients were 0.39, 0.41 and 0.25 for Pack, Unix and ZIP respectively. Thus size is not an indication of a file's susceptibility to being compressed.

Compression

Figure 12 Compression ratio versus file size (Pack)

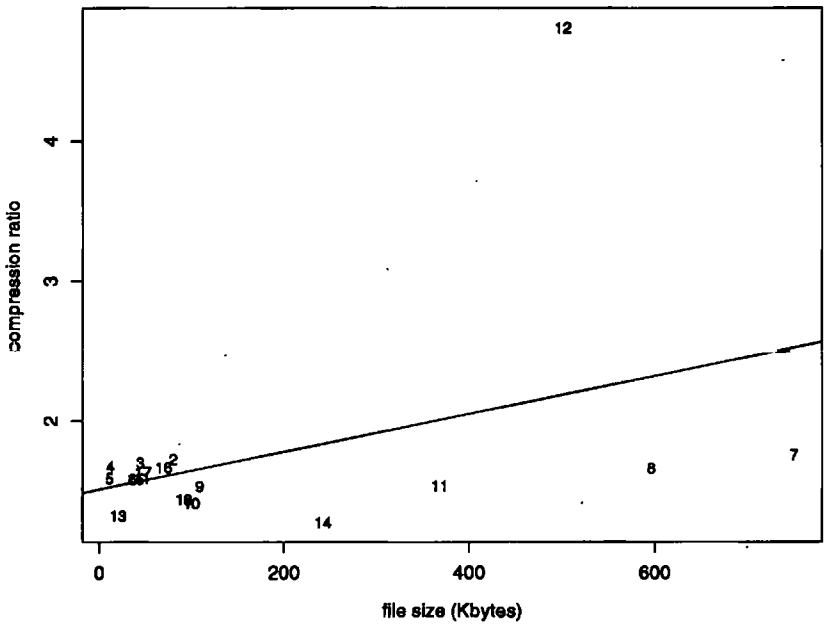


Figure 13 Compression ratio versus file size (Unix)

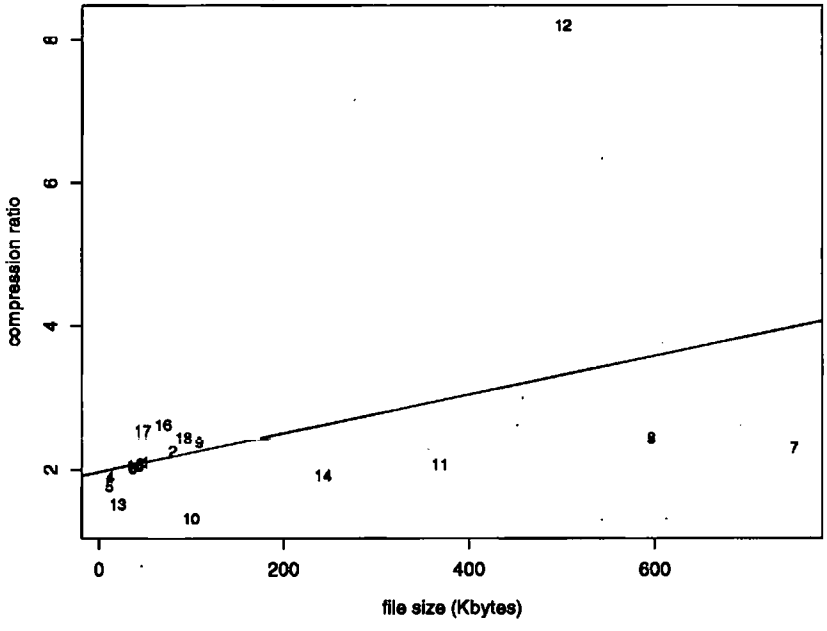
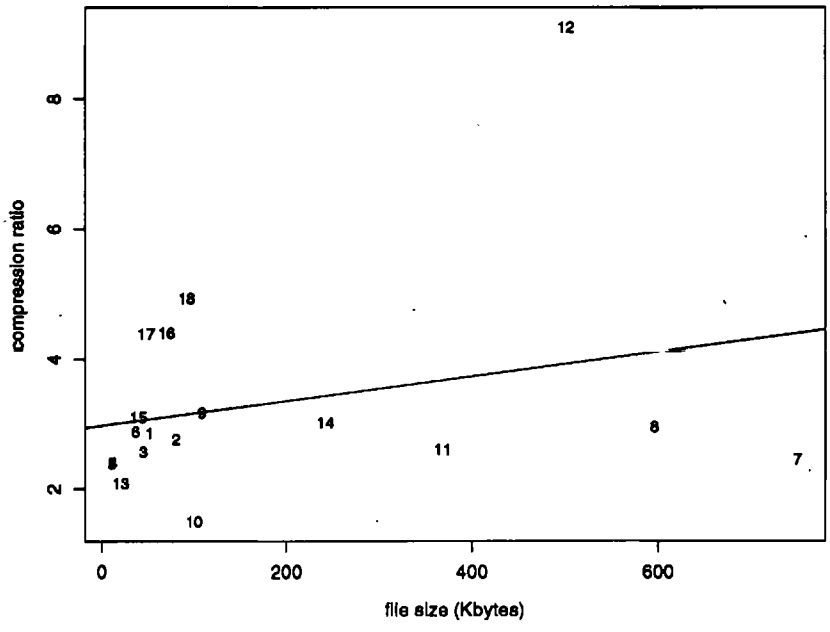


Figure 14 Compression ratio versus file size ZIP



Shannon introduced the concept of Entropy in "A Mathematical Theory of Communications" [214]. Entropy is a measure of information content in a message. Given that there are a total of M different symbols, there exists a set of probabilities P_i that the i^{th} symbol will occur, where $i = 1,2, \ldots M$ and

$$\sum_{i=1}^{i=M} P_i = 1 \tag{13}$$

The *Entropy* E[1] is given by the expression

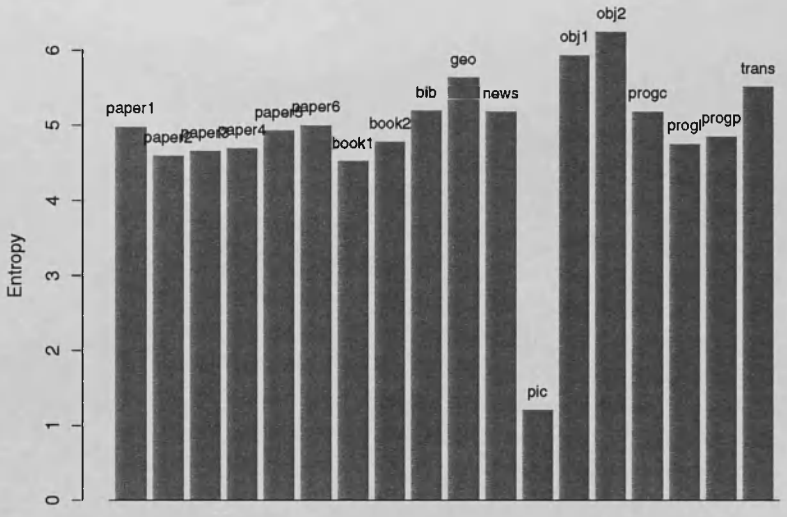
$$E = -\sum_{i=1}^{i=M} P_i \log_2 P_i \tag{14}$$

The probabilities P_i were derived from the relative frequencies of the symbols in each individual file of the corpus suite. Figure 15 shows the value E for each file in the corpus. The file pic has the lowest entropy and therefore contains the least amount of infor-

1. Normally H is used to represent entropy but this conflicts with Hurst parameter H used in CHAPTER 5.

mation, whereas obj2 contains the most amount of information having the highest entropy value.

Figure 15 Entropy



The maximum entropy occurs when $_i = 1/M$ for all i, therefore $E_{\max}$ is

$$E_{\max} = -\sum_{i=1}^{M} \frac{1}{M} \log_2 \frac{1}{M} \tag{15}$$

From this, the average amount of redundancy R in a message can be computed

$$R = E_{\max} - E = \log_2 M + \sum_{i=1}^{M} P_i \log_2 P_i \tag{16}$$

In Figure 16, Figure 17 and Figure 18 each file's compression ratio is plotted against its redundancy R for Pack, Unix and ZIP compression respectively. The correlation are strong, with correlation coefficients of 0.95 for Pack, 0.92 for Unix and 0.78 for ZIP.

Figure 16 Redundancy versus compression ratio (Pack)

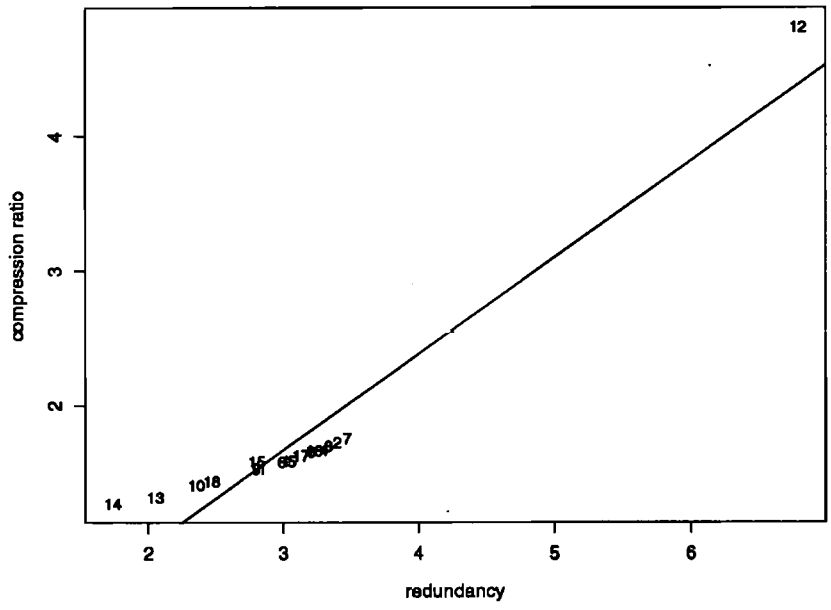


Figure 17 Redundancy versus compression ratio (Unix)

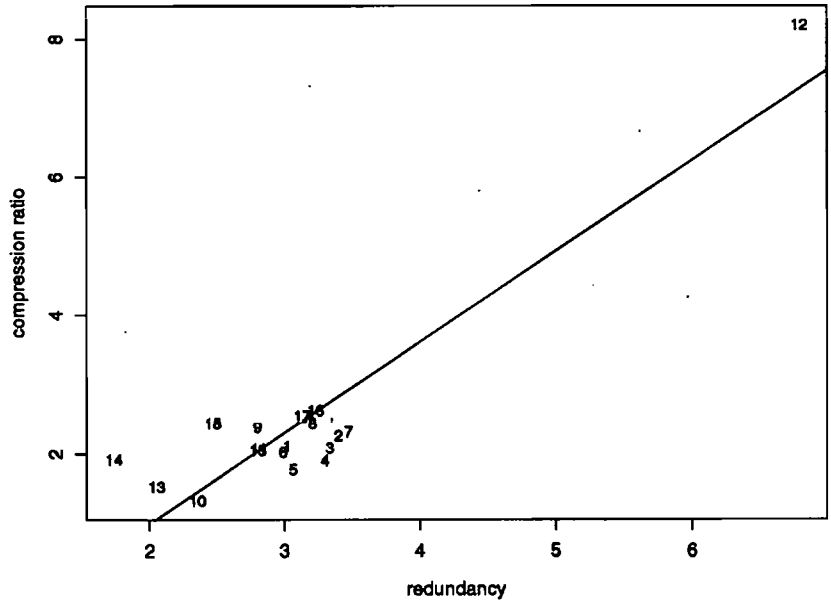
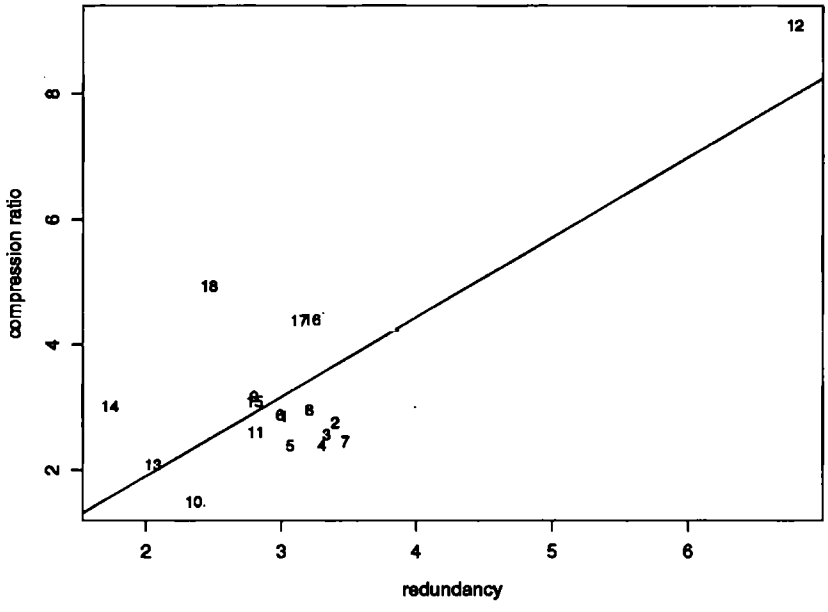


Figure 18 Redundancy versus compression ratio (ZIP)



Redundancy (as derived from Entropy) is not a perfect way to assess the susceptibility of an item of data to be compressed. Furthermore its correlation with compression ratio varies with the method used. It is likely that no single figure of merit exists to represent compression susceptibility for all compression methods. Nevertheless, in section 3.5 it is shown how Redundancy, can be a useful estimate of compression susceptibility.

3.4 Compression Performance

This section describes the results of an empirical study to determine the effects of compression on file transfer speeds within a wide area network environment. In a network environment compression can be done at the application level or in the network itself.

At the application level data is compressed prior to transmission across the network. That is a user can compress a file with a tool like Pack, Unix compress or Zip, or compression is inherent within the encoding scheme such as for image formats.

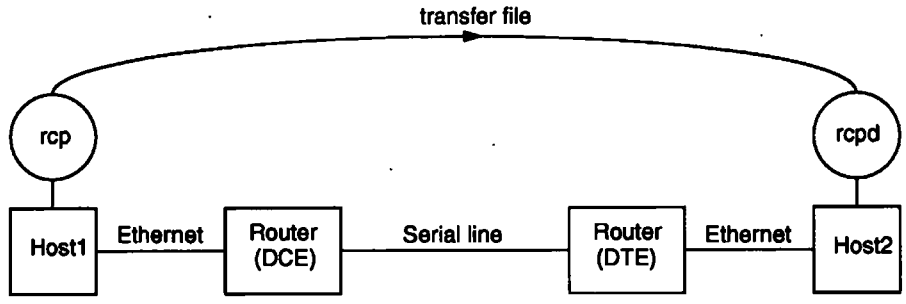
At the network level, routers such as Cisco offer compression on the network interface. As a packet arrives at an interface is it compressed before encapsulating it in the data link frame. On receipt of the frame by the remote interface the data is extracted and decompressed before being routed on.

3.4.1 Experiment

A "wide area network" was set up using two Cisco routers connected via their serial interfaces. One router was configured as a DCE (while the other remained a DTE) so that it could provide the clock for the serial link. Each router in turn was connected to an Ethernet network on which the two test hosts resided.

File transfer performance was evaluated by using `rcp` (Remote Copy) to send each file in the Calgary Corpus suite from one host to the other across the wide area network (see Figure 19).

Figure 19 Experimental environment



The serial link between the two routers was configured for PPP encapsulation (Point-to-Point Protocol) so that data link compression could be enabled.

The control for this experiment consisted of transferring the files without any compression. The time to transfer the files was recorded using the Unix `time` command. For example

```
time rcp trans root@host2:/dev/null
```

This experiment was repeated with serial link compression enabled on the routers. Cisco offers two serial link compression techniques, Predictor and STAC. File transfer performance was measured for both.

Each file was `rcp`ed 15 times and the median value was taken. Care was taken not to send 15 repetitions of the same file back-to-back in case this improved the "prediction" ability of the serial link compression. Instead repetitions of the files were sent "interleaved" with the other file in the corpus suite. That is the algorithm for the bench test script was

```
for i in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
do
    for j in paper1 paper2 paper3 . . .
    do
        time rcp $j root@host2:/dev/null
    done
done
```

rather than

```
for j in paper1 paper2 paper3 . . .
do
    for i in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
    do
        time rcp $j root@host2:/dev/null
    done
done
```

The local files are sent to the `/dev/null` device on the remote host. Data sent to this device is discarded so any disk activity on the remote host is avoided.

Compression performance over the wide area network was also measured using the file compression techniques: Unix and ZIP (with serial link compression on the routers disabled). The time was taken to compress each file, `rcp` it and then uncompress it, for example

```
time (compress trans;
    rcp trans.Z root@host2:/dev/null;
    uncompress trans.Z)
```

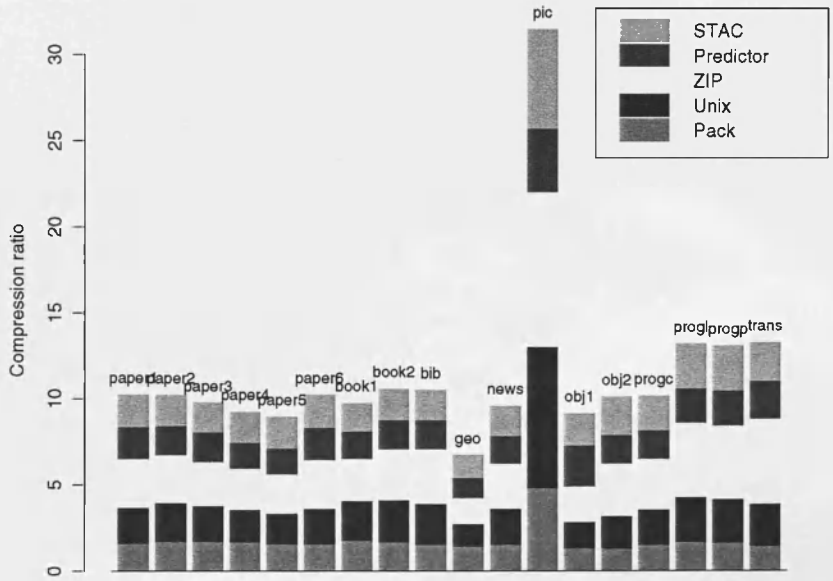
3.4.2 Results

When a file is transferred across the serial link some protocol overhead is added to the data in the file itself. This has an effect on the compression ratio. So for each compression technique, For Predictor, STAC, Unix and ZIP, the compression ratio is given by the expression below rather than the definition in (10)

$$\text{compression ratio} = \frac{\text{no. bytes transmitted sending the non-compressed file}}{\text{no. bytes transmitted sending the compressed file}} \quad (17)$$

The numerator and the denominator in the right hand side of (17) can be obtained from the serial link interface statistics of the router. Figure 20 shows the *transmission* compression ratios for each technique.

Figure 20 Transmission compression ratios for Unix, ZIP, Predictor and STAC



For the file compression techniques, Pack, Unix and ZIP, the average compression ratios over the whole corpus suite were 1.77, 2.39 and 2.93 respectively. It can be seen that

due to the protocol overhead the transmission compression ratios for both these techniques were lower than their file transfer compression ratios.

The packet level compression techniques, Predictor and STAC achieved compression ratios of 1.88 and 2.13 respectively (these results are shown graphically in Figure 21).

Figure 21 Mean compression ratio for entire corpus suite.

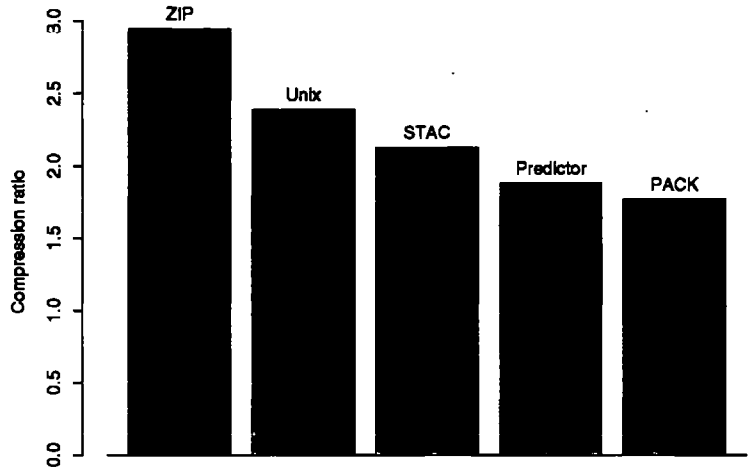


Figure 22 and Figure 23 show compression ratios plotted against redundancy for Predictor and STAC. Compression ratio correlate reasonably well with redundancy (though not very strongly). The correlation coefficients were 0.71 and 0.83 for Predictor and STAC respectively. The correlation between the transmission compression ratio and redundancy for Unix and ZIP differed very little compared to that of their *file* compression ratio results.

Figure 22 Redundancy versus compression ratio (Predictor)

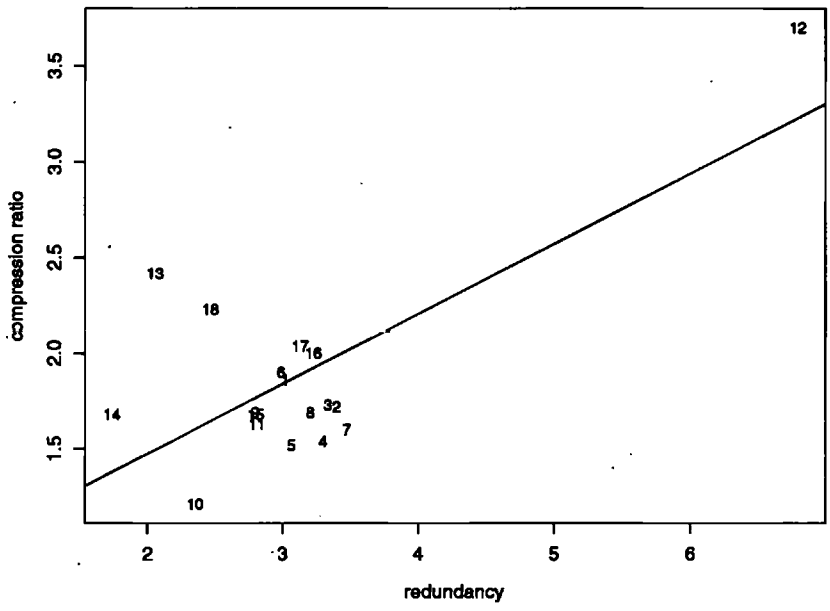
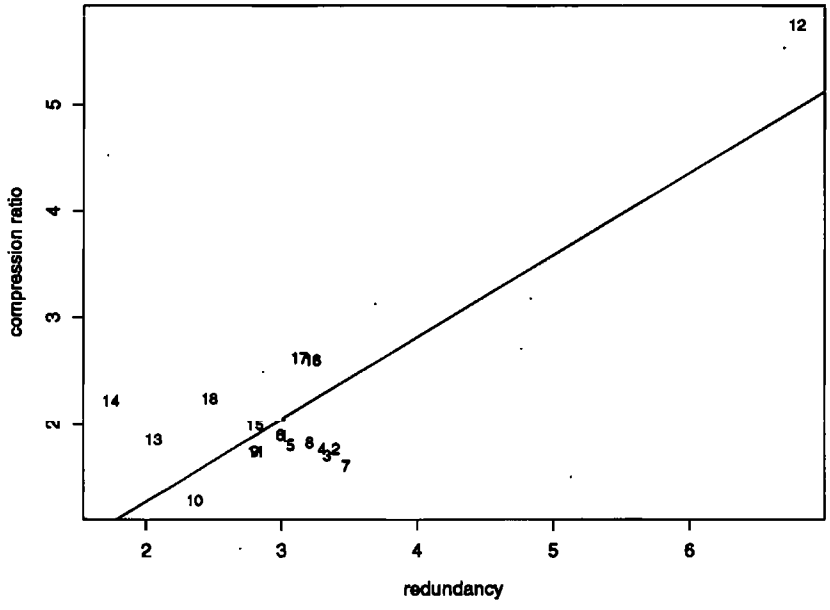


Figure 23 Redundancy versus compression ratio (STAC)



The file transfer time results for each file are shown in the graphs in Appendix B. The graphs show the median time to transfer against bandwidth for each compression tech-

nique. Note that the median was used over the mean in order to counter the effects of outliers. These results are summarised in Figure 24. The per byte delay D_{byte} is computed for each compression technique. Given that for $i = 1,2,...18$, T_i is the median transfer time for each file in the corpus (that is "paper1" = 1, "paper2" = 2,... "trans" = 18) and M_i is the size of each file in the corpus, then

$$D_{\text{byte}} = \frac{\sum_{i=1}^{i=18} T_i}{\sum_{i=1}^{i=18} M_i} \tag{18}$$

An attempt was then made to fit D_{byte} to the following model

$$D_{\text{byte}} = k \times \text{bandwidth}^{-\alpha} + c \tag{19}$$

Using the non-linear modelling functions in S-Plus [45] the parameters k, α and c were estimated. The solid lines in the graph of Figure 24 represent the model given by (19) parameters in Table 7.

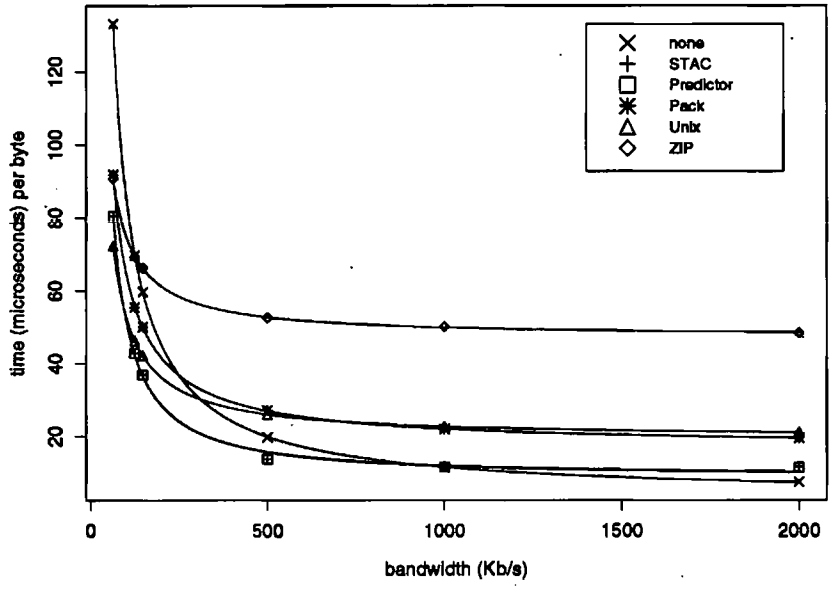
Table 7. Model parameters

	$k \times 10^{-3}$	α	$c \times 10^{-6}$
none	8.17	1.0	3.02
Pack	4.41	0.98	16.91
Unix	3.54	1.01	19.35
ZIP	2.74	1.0	47.06
Predictor	8.06	1.13	8.52
STAC	8.26	1.14	8.79

It can be seen that (19) represents a good fit of the experimental data. Furthermore a direct comparison of compression techniques can be made. Packet level compression (Predictor and STAC), while generally having a better compression ratio, had a lower delay than file level compression (Unix and ZIP). Given that α is close to 1 in each case

$$D_{\text{byte}} \approx \frac{k}{\text{bandwidth}} + c \tag{20}$$

Figure 24 Time/byte versus serial link bandwidth



The reciprocal of D_{bytes} gives these results in terms of a measure of productivity

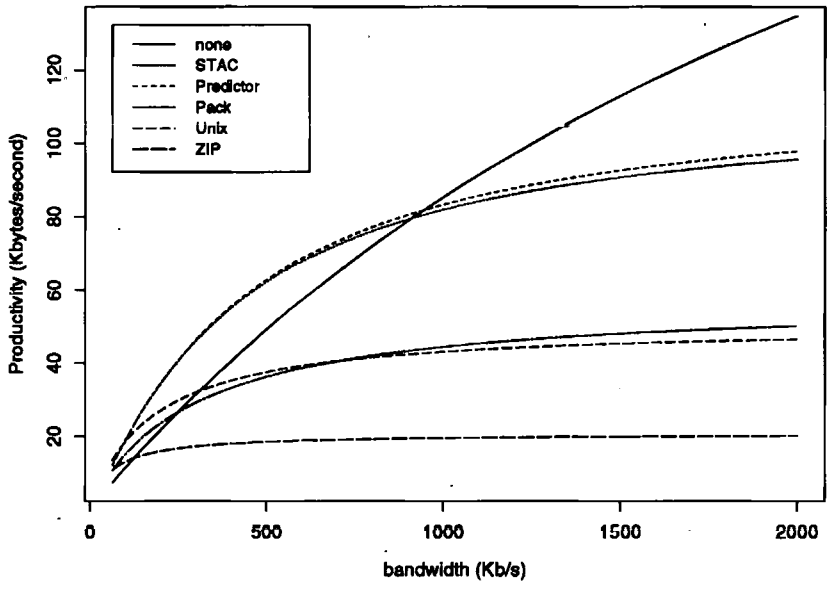
$$P = \frac{1}{D_{\text{byte}}} \tag{21}$$

The graph in Figure 25 shows the productivity results for each compression method. It can be seen that compression is only beneficial for lower link speeds. In general it would appear that doing compression in the network yields higher a productivity curve than doing it at the application level.

STAC and Predictor are very similar to each other in performance as were Pack and Unix. Despite having the highest compression ratio ZIP had the lowest productivity curve.

For Predictor and STAC the compression process starts to become the bottleneck as the link speed increases beyond 1024kb/s. For Pack Unix and ZIP the threshold is much lower, such that compression is not worth while beyond a transmission rate of 128k/bs.

Figure 25 Productivity



3.5 Chapter Discussion

The results in this chapter showed that doing the compression in the network brought about better performance improvements than at the application. However these results should not be taken at face value, they are specific to the environmental conditions under which the experiment was carried out. It is not easy to compare the relative computing power of the routers and the end hosts used in the compression performance experiments. Had end hosts with greater computing power been used, the results may have favoured application compression rather than network compression.

Indeed Intel corporation who develop the microprocessor for PCs claim that "automatic background compression" will soon come about as a result of "higher-performing systems" [120].

Similarly, high specification routers will enhance compression processing times in the network. The router manufacturer Cisco offers a coprocessor module dedicated to packet compression [50], thereby leaving the primary processor to carry out functions concerned with packet forwarding. When compression is done by the router's primary processor it is called software compression (by Cisco) and when it is done by a dedicated module is it called hardware compression.

The results of the experiments carried out in this chapter highlighted that there is a limit to the effectiveness of compression. There is a point where the speed of the communication facility can forward packets faster than packets can be compressed. That is the processor doing the compression becomes the bottleneck rather than the communications facility.

The experiments described in this chapter used software compression. Consequently network compression was not worth doing beyond 512kb/s to 1Mb/s (this value was even lower for compression at the application level). However, with Cisco's dedicated coprocessor, compression is more effective for higher speed communication facilities [50]. Indeed in the Lucent Technologies' network, packets are compressed over 2Mb/s facilities using Cisco's dedicated coprocessor modules.

The choice of where to do compression (application or network) will be influenced by processing resources. However there are other factors to take into account. It should be understood that the conditions under which these experiments were carried were somewhat ideal. The router compression schemes (Predictor and STAC) are *adaptive*, that is the data stream transmitted over the communications facility is analysed to produce a

model. Furthermore the model is continually updated to reflect changes in the data. In the experiments the test files were sent sequentially, this allows the model to adapt to suit each file in order to achieve the optimum compression ratio. However in reality the transmitted data stream will contain packets from a number of simultaneous file transfers. So instead of the model adapting to each individual file (as in the experiments) it will adapt to the data stream which is an aggregate of a number files.

Over the entire corpus of files an average compression ratio of 2.13 was achieved for STAC compression. Examination of one of Lucent Technologies' routers revealed an average compression ratio (for STAC) to be only 1.48.

At the application level compression can be done on a per file basis. The advantage of this (over compressing in the network) is that compression schemes and models can be selected appropriate to the data type in each individual file.

Network compression could be more effective if the router could choose which packets to compress (and which not) and what model or scheme to use. The current trend towards providing quality of service (QoS) in the Internet is to give routers the ability to recognise *flows* in the aggregate data stream [191]. A flow is defined as "a unidirectional sequence of packets between endpoints" [48] specified by

- Source and destination address
- Source and destination port
- Protocol type
- Input interface

Different flows are given different levels of service (according to their requirements). This technology could be used for network compression. Instead of using one model for the entire data stream, there could be a model for each flow. Then each model could

adapt independently to each flow's data. Furthermore the router could select a compression scheme depending upon the type of data in the flow.

Some data types are not susceptible to compression, for instance data that has already been compressed, or certain forms of encrypted data. In section 3.3 entropy was used to assess a file's susceptibility to compression. This same technique could be used at assess a flow's susceptibility. However it is impractical to analyse the entire flow of data in order to determine its entropy value, nevertheless the entropy value can be estimated for the initial bytes of data.

The entropy value for the file paper1 is 4.60. Calculating the entropy of the first 1024 bytes of paper1 gave a value of 4.92. A compressed version (using the Unix Compress command) had an entropy value of 7.99 (the maximum entropy value being 8.00). The entropy value of the first 1024 bytes was 7.79. Similarly paper1 was encrypted, yielding an entropy value of 7.95 and an entropy value of 7.64 for the first 1024 bytes.

It has been shown that the data in the first packet is sufficient to give a reasonable estimate of the entropy of an entire flow. Data with entropy value close to 8.00 (such compressed and encrypted files) cannot be compressed, so the router can elect to not compress flows with entropies close to this value.

Compression could also be applied on priority basis. Flows with a low priority (provided they are susceptible) could be compressed while high priority flows are not. This would be particularly useful if processing resources for doing compression are limited. Low priority flows will incur additional delays awaiting compression processing to complete, while high priority flows will not. Furthermore high priority flows will benefit from reduced delays due to the reduction in traffic volumes.

In game theory terms some players (low priority flows) are forced to "cooperate" and reduce their load on the network in order to benefit the "defectors" (high priority flows). Typically the roles of cooperators and defectors will be determined by ability to pay.

Another opportunity to apply compression is when the network congested. During periods of congestion, packets spend greater time waiting in queues. This time could be used to carry out compression on queued packets, which would reduce queue lengths and so avoiding packets being dropped.

3.6 Chapter Summary

A non-linear model of the performance of each compression method against link speed was fitted to the results of the experiments carried out in this chapter. This provided a useful means of comparing the relative performance of each method.

Nevertheless the decision to use compression to enhance network performance is a complex one. The effectiveness of compression is dependent upon the processing resources available. The results of the compression performance experiments have shown that the compression process itself can be a bottleneck if CPU capabilities are limited. Improving network performance then becomes a trade-off between incurring the cost of adding processing resources (to routers and end hosts) and upgrading bandwidth.

There is also the issue of what is the optimum compression ratio. While schemes that deliver high compression ratios bring about greater reductions in data, the processing requirements of these schemes may be beyond the processing power available. Some schemes achieve high compression ratios by incurring losses in the original data (for example JPEG). Such schemes may not be suitable if losses are deemed too high. In cases where performance and quality are an issue, lower compression ratios may be more desirable

Chapter Summary

Wholesale compression of packets transmitted over communications facilities is not effective because aggregated streams of data from homogeneous sources are not as susceptible to compression as the individual units of data. When compression is done at the application level, schemes and models can be used appropriate to the data type. The problem with application compression is that the decision to compress is given to the user. While an individual user's decision to compress data will benefit other users of the network due to reduced queue lengths, the compressing user may incur higher delays if processing power is insufficient. In which case a user's choice to compress data is purely an altruistic one. It is therefore reasonable to assume that, despite the overall benefits, network engineers cannot rely on compression at the application level to bring about network performance enhancements.

Flow technology, offers a means of doing compression selectively. Flows can be compressed based upon susceptibility, priority and periods of congestion. Therefore compression could be used more effectively if it were included as a Quality of Service feature.

CHAPTER 4 *Mirrors*

A mirror is a server that replicates the contents (or a substantial fragment of the contents) of another server. However, unlike replicates installed for fault tolerance, mirrors are only updated periodically so at times the mirror is out of step with its original. Remote Web sites vary in popularity. Popular sites can be very busy which can impact performance. Performance bottlenecks on Web servers can be overcome by load balancing document requests across a number of co-located servers. However busy sites can also cause network *hot spots*. By locating (popular) sites at different locations some of the load can be distributed to different parts of the network so that "overall" queuing delays are reduced. Also if users request files from a mirror that is closer than the original remote server the propagation delays incurred will be less.

This chapter investigates the use of mirrors as a means of improving network performance within an enterprise. The productivity of a network user is affected by the amount of time spent waiting for documents to download from a remote server. The model (8) introduced in CHAPTER 2 showed the component network delays. A mathematical analysis, using (8), of the performance experienced by a population of users accessing a remote server is presented. The purpose of this analysis is to reveal the dilemmas presented in section 2.3 of CHAPTER 2. The analysis is developed further to show the

performance of a remote server with a mirror site. The resultant analytical model provides a means of investigating the effects of mirror location and the speed of communication facilities.

One of the inherent problems with replication is the risk of accessing stale data (data that has changed on the original server but not on the mirror). Users may refrain from using mirrors (in favour of the original server) if they suspect there is a high risk of accessing stale data. The stale data problem can be reduced by increasing the frequency of document updates. However this can increase load on the network. The relationship between the risk of stale data and update frequencies are analysed in this chapter.

The benefits of replication were demonstrated in an experiment using a distributed application used by software developers in Lucent Technologies called Teamwork. Despite developers being located at different sites it was necessary for them to share data generated by the Teamwork application. An analysis of the way Teamwork accessed data over the network revealed that replication could be utilised in this environment. The experiment showed that replication reduced access delays and traffic volumes over the wide area network.

Techniques such as replication will not necessarily bring about performance improvements if the "model" of the way users access data is inappropriate. This experiment illustrated the importance of understanding how users access data within a particular application.

4.1 Web Site Popularity

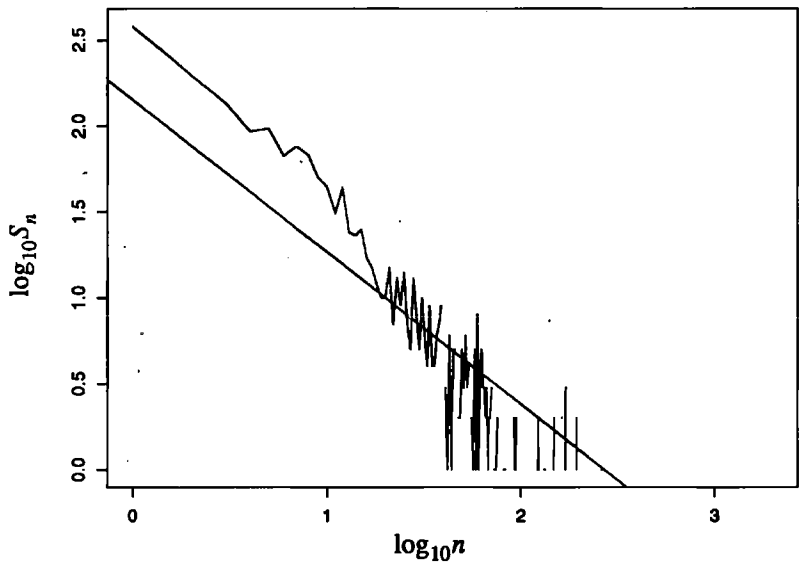
The graph in Figure 26 shows the distribution of accumulated requests across different sites based upon data collected from an actual proxy server.

Given that S_n is the number of servers that have accumulated n references then the following relationship can be derived from the fitted line shown in Figure 26

$$S_n = 141 \times n^{-0.89} \quad (22)$$

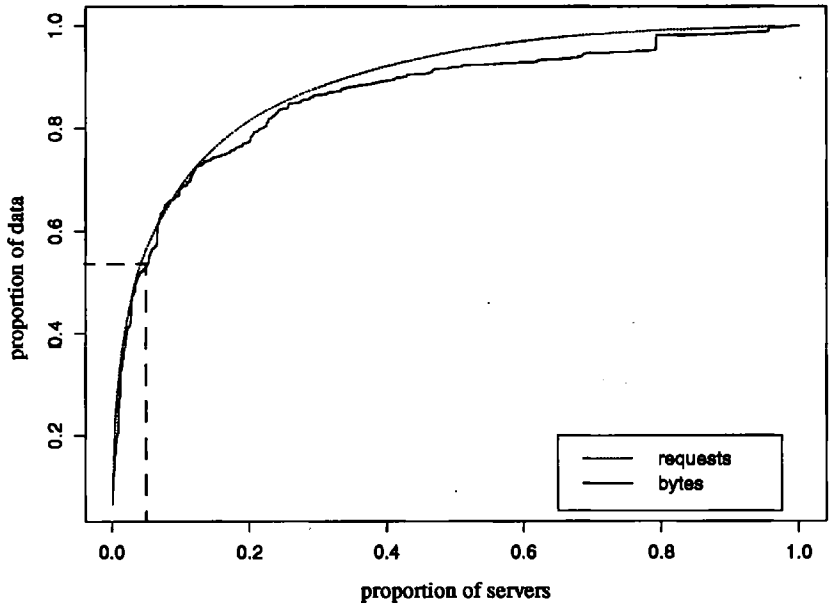
This analysis shows that the number of accumulated requests are unevenly distributed amongst servers. That is fewer sites are accessed frequently than are accessed infrequently.

Figure 26 Popularity of Web sites



The graph in Figure 27 shows the proportion of servers versus the accumulated proportion of data. The majority of data is supplied by relatively few servers, therefore it can be reasoned for the graph in Figure 27 that mirroring 5% of the servers closer to the users would reduce the data on the wide area network by just over 50% (see dashed line in Figure 27).

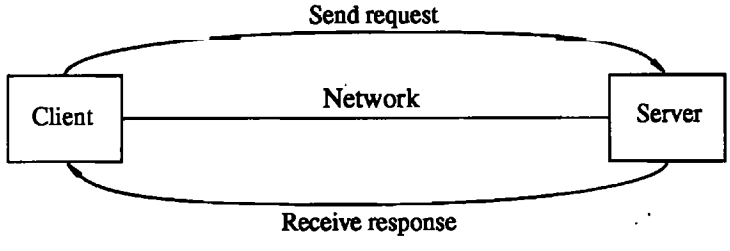
Figure 27 Proportion of data versus proportion of servers



4.2 Remote Server Analysis

The communication between a client and a server is typically in the form of a request-response exchange of messages. Using the World-Wide Web as an example, when a software client wants to download a particular document (HTML page, GIF image or MPEG video sequence) it sends a request message to the server. On receiving the request the server sends the contents of the file in which the document is stored.

Figure 28 Requesting a document from a server



4.2.1 Round Trip Delay

The round trip delay is the time it takes for the client to send a request to the server and receive a reply (the requested resource). The model for packet delay (8) in CHAPTER 2 is used to develop a model for the round trip delay in requesting a document from a remote server. The round trip delay D_{RT} is given by

$$D_{RT} = D_{L_S} + D_{L_R} \tag{23}$$

where D_{L_S} is the time it takes to send the request from the client to the server and D_{L_R} is the time it takes to receive the response from the server to the client. For each direction, client-server and server-client, the respective latencies can be expressed

$$D_{L_S} = D_{C_S} + D_{W_S} + D_{F_S} + D_P \tag{24}$$

and

$$D_{L_R} = D_{C_R} + D_{W_R} + D_{F_R} + D_P \tag{25}$$

where,

- D_{C_S} and D_{C_R} are the respective client-server and server-client processing delays
- D_{W_S} and D_{W_R} are the respective client-server and server-client waiting delays
- D_{F_S} and D_{F_R} are the respective client-server and server-client framing delays
- D_P is the propagation delay

The respective queuing delays are given by

$$D_{Q_S} = D_{W_S} + D_{F_S} = \frac{1/\mu_S}{1 - k_S n} \tag{26}$$

and

$$D_{Q_R} = \frac{1/\mu_R}{1 - k_R n} \tag{27}$$

where n is the number of currently active users. k_S is the average client-server network traffic work load per user and k_R is the average server-client network traffic work load per user where and $0 \leq k_S < 1$ and $0 \leq k_R < 1$.

The service rates μ_S and μ_R are given in terms of messages per second. Although, it is assumed, the bandwidth of the network between client and server is the same in both directions, the send and receive message sizes will be different. Typically requests for documents from clients are small and may comprise a single packet of data whereas the response from the server can be quite large (a whole document). Typically therefore $\mu_S > \mu_R$. Given that M_S is the average size of a request sent by the client and M_R is the average size of the document returned by the server

$$\mu_S = \frac{B}{8 \times M_S} \tag{28}$$

and

$$\mu_R = \frac{B}{8 \times M_R} \tag{29}$$

Where M_S and M_R are in bytes, and B is the bandwidth in bits per second of the symmetric network link between client and server. The propagation delay is given by

$$D_P = \frac{d}{c_F} \tag{30}$$

where d is the distance between client and remote server (along the path of channel) and c_F is the speed of light through fibre (180,000 km/s according to [191]). For the purpose of this analysis it is assumed that the router processing delay is negligible ($D_C \approx 0$). Thus

$$D_{RT} = D_{Q_S} + D_{Q_R} + 2D_P \tag{31}$$

Substituting (26), (27) and (30)

$$D_{RT} = \frac{1/\mu_S}{1-k_S n} + \frac{1/\mu_R}{1-k_R n} + \frac{2d}{c_F} \tag{32}$$

Multiplying out gives

$$D_{RT} = \frac{1/\mu_S(1-nk_R) + 1/\mu_R(1-nk_S) + \frac{2d}{c_F}(k_R k_S n^2 - nk_R - nk_S + 1)}{k_R k_S n^2 - nk_R - nk_S + 1} \tag{33}$$

4.2.2 Productivity

A user's productivity diminishes as the time spent waiting for documents to download increases. The data rate in messages per seconds generated by an individual user is inversely proportional to the round trip delay D_{RT} and is given by

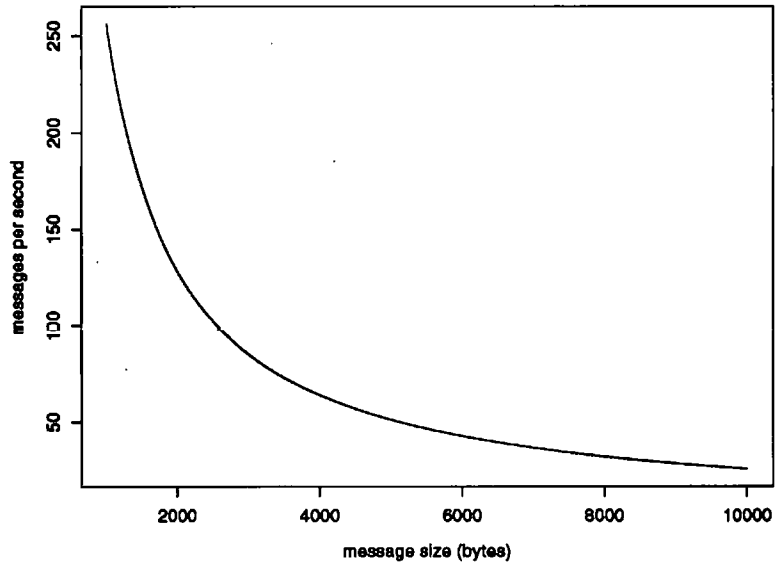
$$\frac{k_R k_S n^2 - nk_R - nk_S + 1}{1/\mu_S(1-nk_R) + 1/\mu_R(1-nk_S) + \frac{2d}{c_F}(k_R k_S n^2 - nk_R - nk_S + 1)} \tag{34}$$

The aggregate data rate for the system is the product of the date rate for an individual and the number of users of the network n, that is

$$\frac{n(k_R k_S n^2 - nk_R - nk_S + 1)}{1/\mu_S(1-nk_R) + 1/\mu_R(1-nk_S) + \frac{2d}{c_F}(k_R k_S n^2 - nk_R - nk_S + 1)} \tag{35}$$

As the services rates μ_S and μ_R are in units of messages per second the expression (35) gives the productivity in messages per second. Figure 29 shows a graph of the productivity of transmitting a message over a communications facility versus message size.

Figure 29 Data rate (messages per second) versus message size.

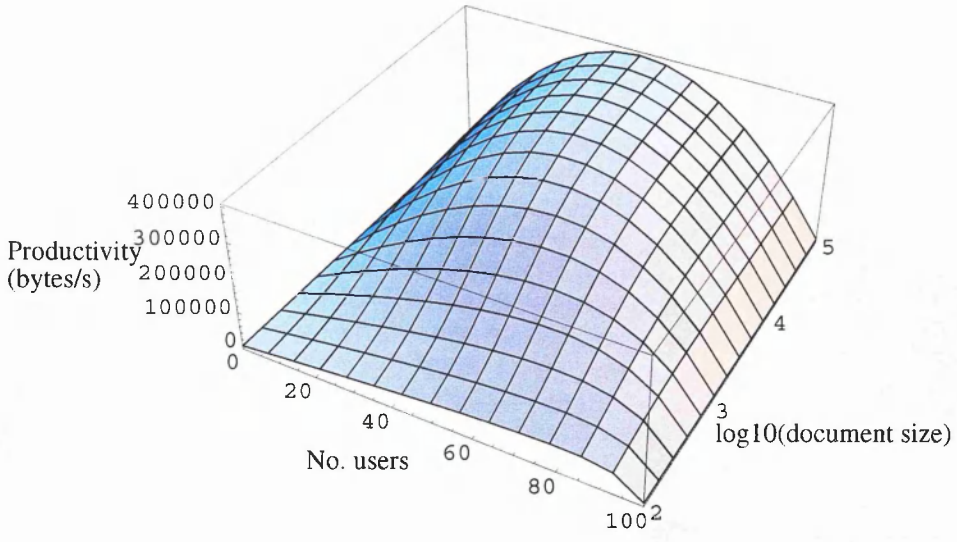


It can be seen that as the message size increases the productivity *appears* to decrease. This is because it takes longer for a communications facility to transmit long message than short messages. However we would also expect a large document to contain more information and this is not accounted for in the current calculation. For this reason the (35) is multiplied by the average message size M_R to give an expression for productivity in bytes per second

$$P = \frac{nM_R(k_R k_S n^2 - nk_R - nk_S + 1)}{1/\mu_S(1 - nk_R) + 1/\mu_R(1 - nk_S) + \frac{2d}{c_F}(k_R k_S n^2 - nk_R - nk_S + 1)} \quad (36)$$

The graph in Figure 30 shows the productivity (in bytes per second) attained from using a remote server 8000km away (that is $d = 8000$) over a communications link of 128kb/s. Productivity is plotted against the number of users using it (out of a population 100) for different (average) document sizes.

Figure 30 Productivity of using a remote server



The graph in Figure 30 shows the total productivity is influenced by average message size and the number of users. Increases in productivity correspond to increases in document size. Increases in users increase the total productivity until it reaches an optimum value, that is when

$$\frac{dP}{dn} = 0 \tag{37}$$

As the number of users increases beyond this point the productivity decreases because the network starts to become congested. The graph also shows that the value for n for which productivity is an optimum is dependant upon the average document size.

The graph in Figure 31 shows the number of active users n, that are needed to optimise the total productivity.The optimum number of users depends on the mean document size as Figure 31 confirms.

Figure 31 Number of users needed to optimise productivity

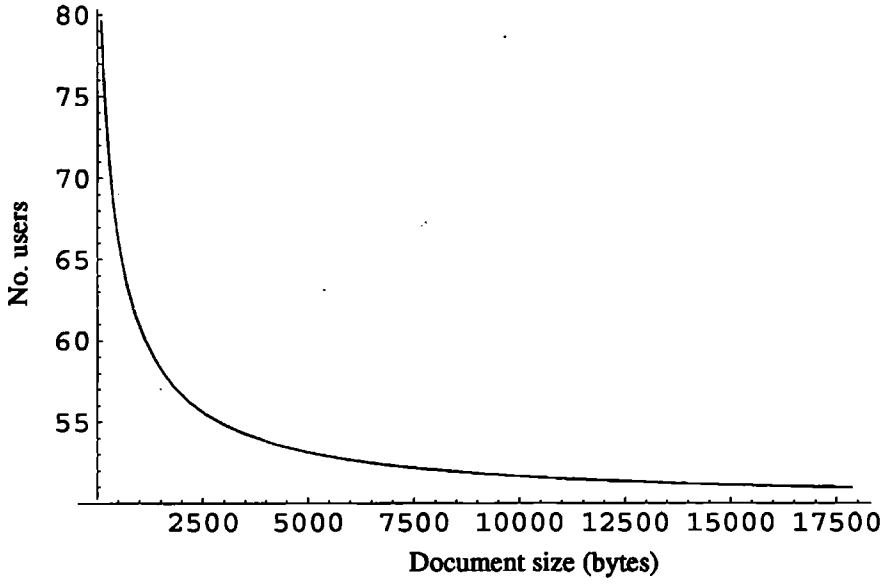
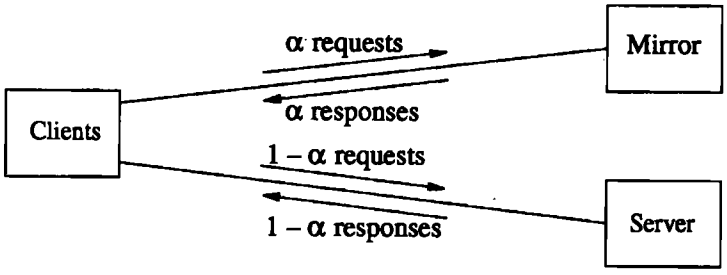


Figure 30 shows that the optimum productivity is better for fewer users and larger document sizes. That is as the mean document size increases (and the productivity in general) the number of users for which productivity is at its optimum value diminishes. This illustrates the dilemma introduced in CHAPTER 2 whereby there is a conflict of interest between the individual user and the "common" network good. In order to maintain "optimum" network productivity, some users must sacrifice their network access. And while productivity increases with mean document size so too does the number of users excluded from using the network if the optimum productivity is to be achieved.

4.3 Mirror Analysis

The productivity model (36) is developed to include a mirror site.The diagram in Figure 32 shows the mirror architecture.

Figure 32 Mirror architecture



From a population of users a proportion α will use the mirror site and a proportion $1-\alpha$ will use the original remote server. The round trip delays incurred fetching a document from the remote server is

$$\frac{1}{(1-k_{S_r}[1-\alpha]n)\mu_{S_r}} + \frac{1}{(1-k_{R_r}[1-\alpha]n)\mu_{R_r}} + \frac{2d_r}{c_F} \tag{38}$$

The data rate for an individual user of the remote is the reciprocal of the delay

$$\frac{1}{\frac{1}{(1-k_{S_r}[1-\alpha]n)\mu_{S_r}} + \frac{1}{(1-k_{R_r}[1-\alpha]n)\mu_{R_r}} + \frac{2d_r}{c_F}} \tag{39}$$

Similarly the round trip delays incurred fetching a document from the mirror server is

$$\frac{1}{(1-k_{S_m}\alpha n)\mu_{S_m}} + \frac{1}{(1-k_{R_r}\alpha n)\mu_{R_m}} + \frac{2d_m}{c_F} \tag{40}$$

and the data rate for an individual user of the mirror is

$$\frac{1}{\frac{1}{(1-k_{S_m}\alpha n)\mu_{S_m}} + \frac{1}{(1-k_{R_r}\alpha n)\mu_{R_m}} + \frac{2d_m}{c_F}} \tag{41}$$

where

- μ_{S_m} and μ_{S_r} are the respective client-mirror, client-remote service rates.

- μ_{R_m} and μ_{R_r} are the respective mirror-client, remote-client service rates
- k_{S_m} and k_{S_r} are the respective average client-mirror, client-remote per user workloads
- k_{R_m} and k_{R_r} are the respective average remote-client, mirror-client per user workloads
- d_m and d_r are the respective distances of the mirror and remote from the client

The aggregate data rate of using a remote server with a mirror is given by

$$R = \frac{n(1-\alpha)}{\frac{1}{(1-k_{S_r}[1-\alpha]n)\mu_{S_r}} + \frac{1}{(1-k_{R_r}[1-\alpha]n)\mu_{R_r}} + \frac{2d_r}{c_F}} + \frac{n\alpha}{\frac{1}{(1-k_{S_m}\alpha n)\mu_{S_m}} + \frac{1}{(1-k_{R_m}\alpha n)\mu_{R_m}} + \frac{2d_m}{c_F}} \quad (42)$$

Assuming the total number of potential users is given by T and the total bandwidth is B then the bandwidth per user is given by

$$\frac{B}{T} \quad (43)$$

The rate of transmission of documents per user is

$$\frac{B}{Tm} \quad (44)$$

Where m is the message size. The total possible traffic in messages per second for message size m is

$$\frac{B}{m} \quad (45)$$

Because the received messages are larger than the requests that are sent, the received message size imposes a limit on the rate at which transactions can be performed. The

proportion of traffic for each user for a stream of messages of size m taking into account this bound is

$$\frac{m}{TM_R} \tag{46}$$

where M_R is the size of the received messages. The proportion of traffic for each user in the send queue at the mirror and the remote server

$$k_{S_r} = k_{S_m} = \frac{M_S}{TM_R} \tag{47}$$

and for the receive queue

$$k_{R_r} = k_{R_m} = \frac{1}{T} \tag{48}$$

With n active users which be can written as a fraction of β of the total number of potential users T

$$n = T\beta \tag{49}$$

Substituting the parameters above into (42) gives the aggregate data rate as

$$R = \frac{T(1-\alpha)\beta}{\frac{M_R}{(1-[1-\alpha]\beta)B_r} + \frac{M_S}{\left(1-\frac{[1-\alpha]\beta M_S}{M_R}\right)B_r} + \frac{2d_r}{c_F}} + \frac{T\alpha\beta}{\frac{M_R}{(1-\alpha\beta)B_m} + \frac{M_S}{\left(1-\frac{\alpha\beta M_S}{M_R}\right)B_m} + \frac{2d_m}{c_F}} \tag{50}$$

In expression (50) B_r is the bandwidth of the communication link to and from the remote server and B_m is the bandwidth of the communication link between the client and the mirror. Values have been substituted into expression (50) the data rate of a system with a remote site and a mirror, where both the remote and the mirror are located

8000km from the client(s), that is $d_r = d_m = 8000$. Also the remote and mirror communication facilitates operate at 128kb/s, that is $B_r = B_m = 128000$. Table 8 gives a summary of the model parameters.

Table 8 Model parameters

Parameter	Value
d_r	8000
d_m	8000
B_r	128000
B_m	128000

The send message size M_S is assumed to be 100 bytes. A value for the receive message size was derived from the Malmesbury proxy Web server. An examination of the log files revealed that the median document size to be approximately 2,000 bytes and was used as the value of M_R throughout by the following analyses.

Productivity is given by the rate of serving documents times the information content of the documents which has been assumed to be related to the received document size. Performing calculations with a fixed average received document size throughout makes productivity proportional to the data rate. The following graphs therefore show the data rates which are uniformly scaled versions of the productivity curves.

The graph in Figure 33 shows the aggregate data rate in messages per second plotted against the proportion of users on the network β and proportion of those users using the mirror α. The graph in Figure 34 shows cross sections of the productivity surface in Figure 33 for various values of β plotted against α.

The aggregate data rate is low when β is small. It increases to an optimum as β increases and where $\alpha = 0.5$ (that is the users are load balanced equally between the remote and the mirror). However for values of $\alpha \to 0$ and $\alpha \to 1$ the aggregate data rate decreases for large values of β. When $\alpha = 0$ (all users using the original server)

or $\alpha = 1$ (all users using the mirror), the aggregate data rate curves are like those for the "no mirror" case.

Figure 33 Aggregate data rate

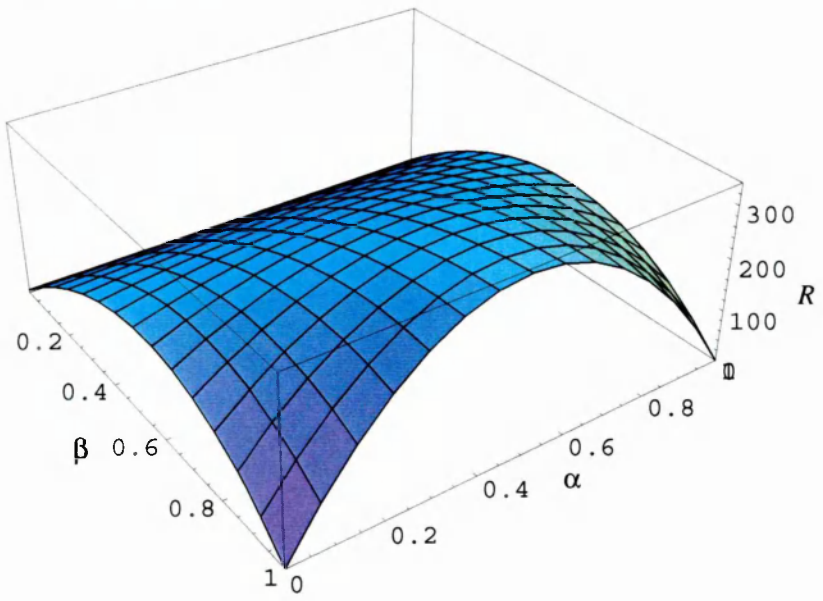
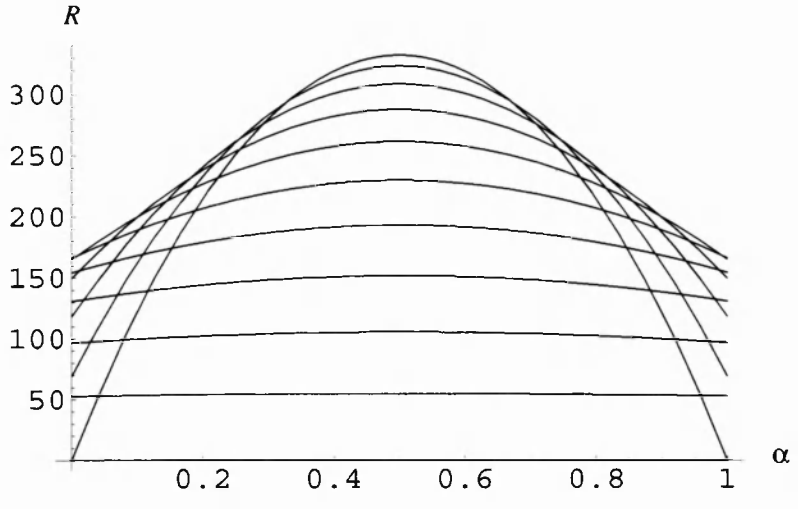
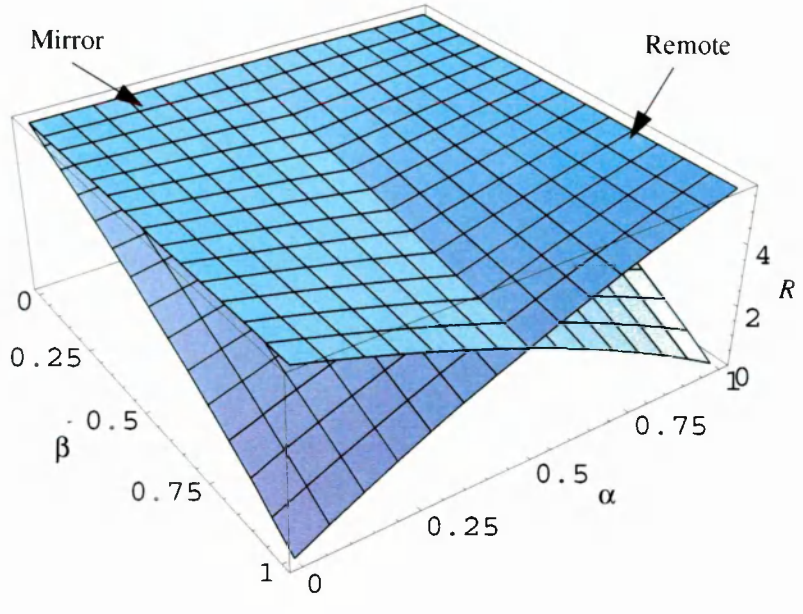


Figure 34 Cross section of aggregate data rate surface



The graph in Figure 35 shows the individual data rate surfaces for both remote and mirror servers. For low values of β the data rate surfaces are high and relatively flat for both the remote and mirror. The surface of the mirror data rate declines as α and β both increase, that is as the mirror gets busy. The same happens for users of the remote as β increases and α decreases and the remote gets busy.

Figure 35 Data rate of individual remote and mirror users



The analytical delay model was used to calculate the component delays (framing delay, propagation delay and waiting delay) for a user of the remote server against α (for $\beta = 0.8$). These results are shown in the graph in Figure 36.

It can be seen that framing delay is comparatively small. Propagation delay is more significant but the overall delay is dominated by the waiting delay. The waiting delay for a remote user decreases as some users migrate to the mirror ($\alpha \rightarrow 1$). The component delays are similar for the mirror users (Figure 37), except that waiting decreases as users migrate to the remote ($\alpha \rightarrow 0$).

Figure 36 Component delays for the remote server

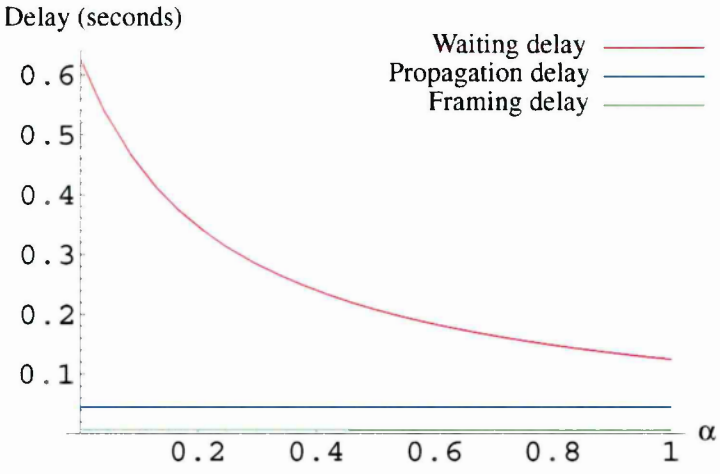
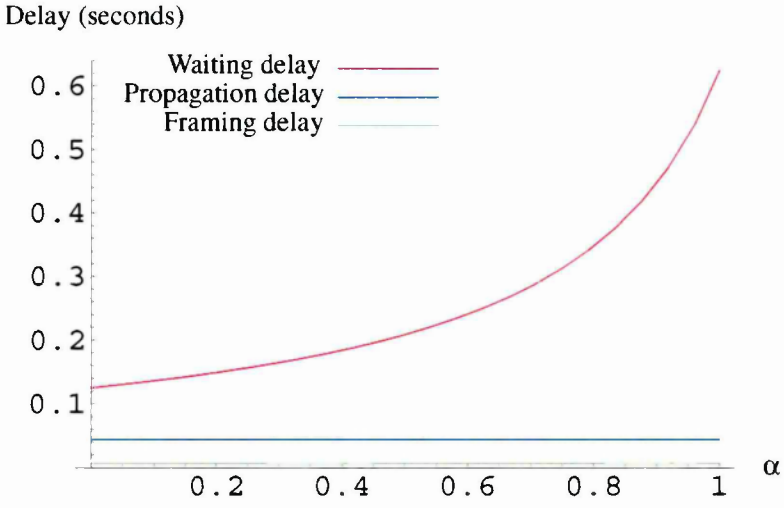


Figure 37 Component delays for the mirror

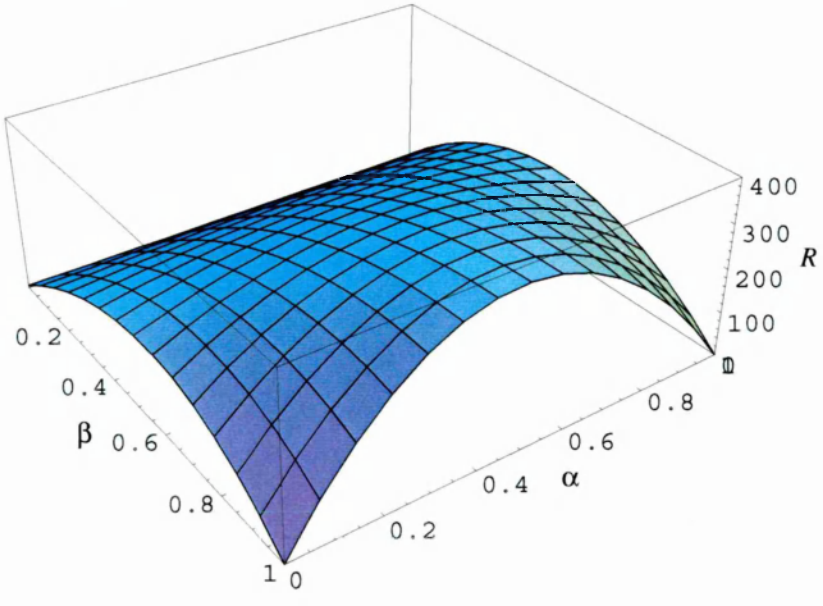


The next case show the effect on data rate of bringing the mirror closer to the client(s). The graph in Figure 38 shows the data rate for using a remote server located 8000km from the clients with a mirror site located 10km away. The remote and mirror communication facilities remain at 128kb/s. Table 9 gives a summary of the model parameters.

Table 9 Model parameters

Parameter	Value
d_r	8000
d_m	10
B_r	128000
B_m	128000

Figure 38 Aggregate data rate



There is some small improvement in the aggregate data rate as a result of bringing the mirror closer. Also the surface is no longer symmetrical about $\alpha = 0.5$. Though this is not easy to see in Figure 38, the cross sectional graph in Figure 39 reveals more clearly

a slight *skew*, such that the optimum for the aggregate data rate occurs when α is just less than 0.5.

Figure 39 Cross section of the aggregate data rate surface

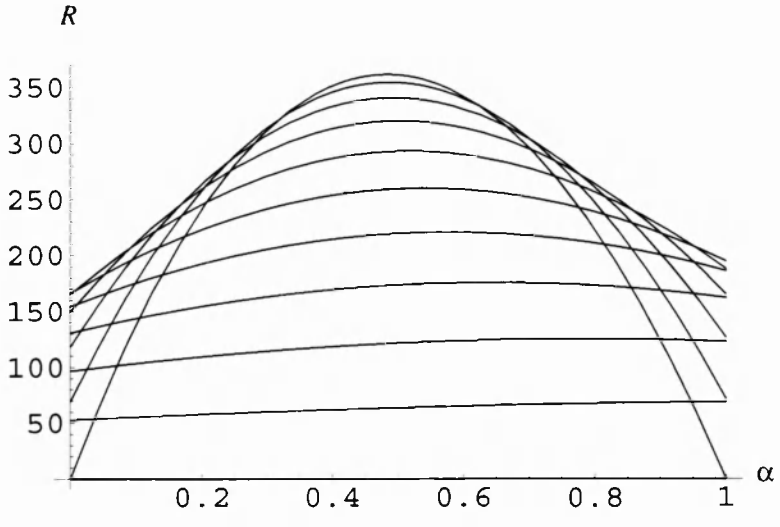
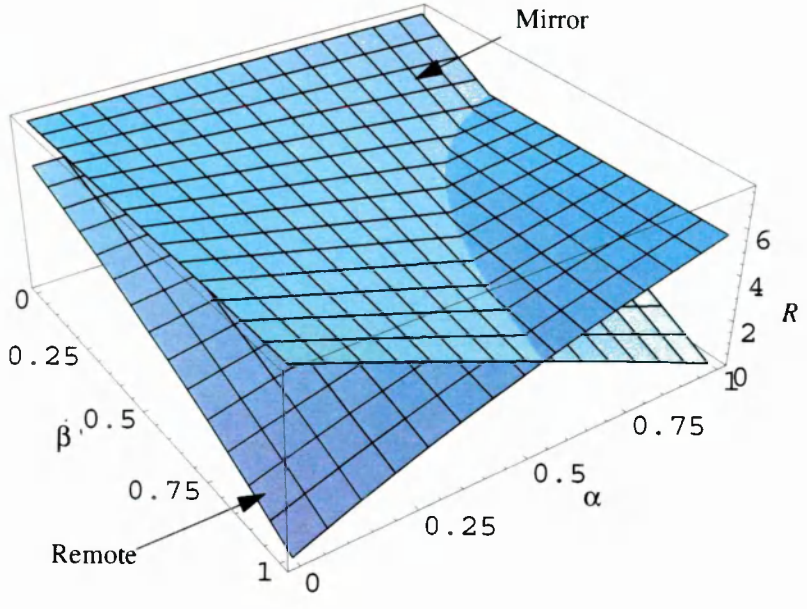


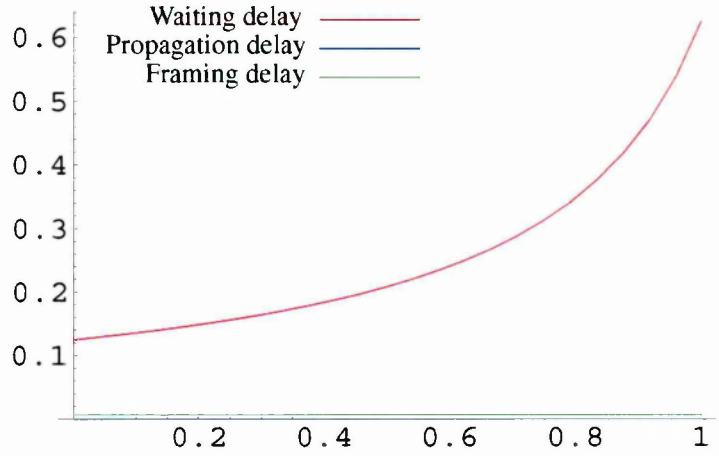
Figure 40 shows the date rate surface for both the individual remote and mirror users. The surfaces are similar to those in Figure 35, however, the mirrors expected tends to yield a higher data rate than the remote.

Figure 40 Data rate of individual remote and mirror user



The component delays for users of the remote server are same as those shown in Figure 36. For users of the mirror though it can be seen by comparing Figure 36 with Figure 41 that the propagation delay is less for the mirror than the remote. Nevertheless, in both cases the overall delay is still dominated by the waiting delay which explains why the gains in data rate are small as a result of bringing the mirror closer.

Figure 41 Component delays for the mirror



An advantage of bringing the mirror closer to the client (reducing d_m) is that short range communication facilities are less expensive. The graph in Figure 42 shows the aggregate data rate surface and Figure 43 shows the cross sectional view with the communication facility between the client and the mirror upgraded to 1.024Mb/s. Table 10 gives a summary of the model parameters.

Table 10 Model parameters

Parameter	Value
d_r	8000
d_m	10
B_r	128000
B_m	1024000

Figure 42 Aggregate data rate

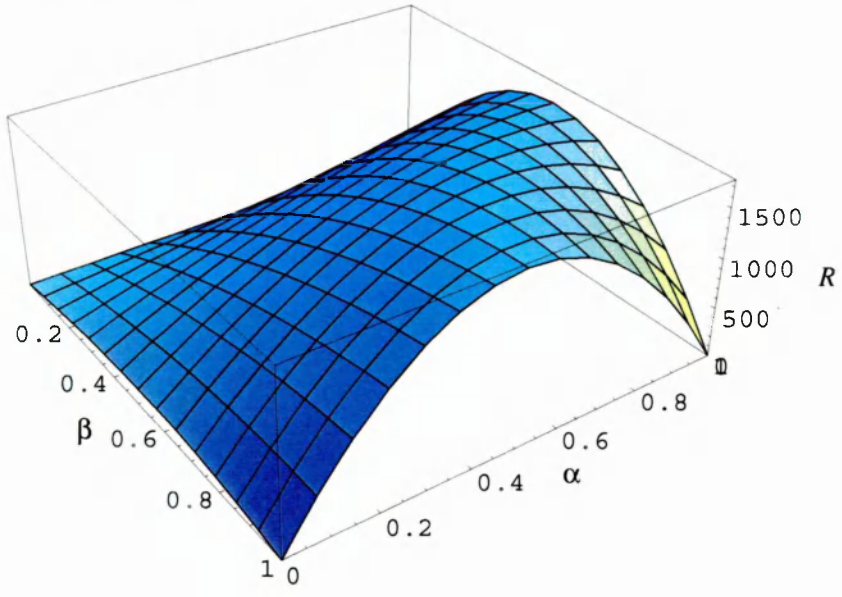
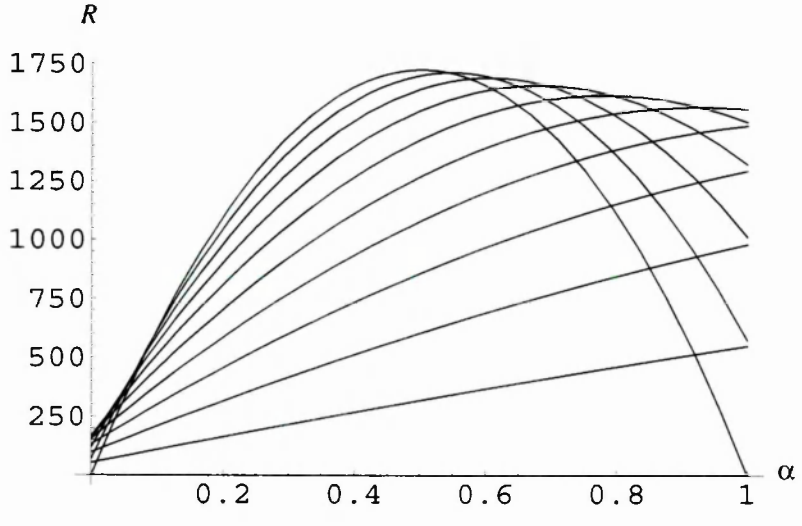


Figure 43 Cross section of the aggregate data rate surface



The graph in Figure 42 (and Figure 43) shows there is a significant increase in the data rate due to the upgrade of the communications facility to the mirror. For values of $0 < \beta < 1$ the data rate is significantly higher when $\alpha \to 1$ (users migrate to the mirror) than for $\alpha \to 0$ (users migrate to the remote).

Figure 44 shows the data rate surfaces for the individual remote and mirror users. It shows that except for values of α and β close to 1 users of the mirror will be more productive than users of the remote).

Figure 44 Data rate of individual remote and mirror user

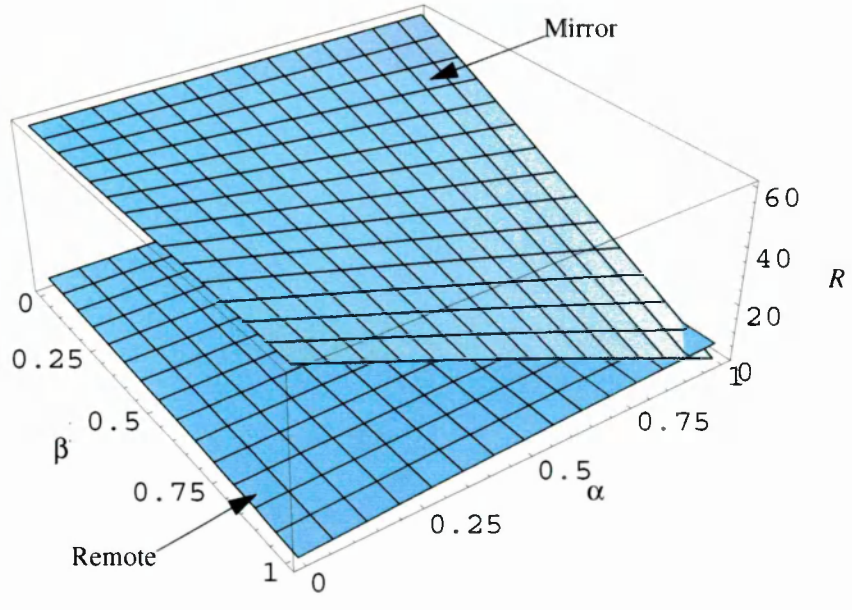


Figure 45 shows the component delays for the mirror site (at 1.024Mb/s). Framing and waiting delays are much lower than for the case with the communication facility to the mirror at 128kb/s. Nevertheless waiting delay is still highest contributor to the overall delay. Propagation delay as expected is unaffected by bandwidth upgrades.

Figure 45 Component delays for the mirror

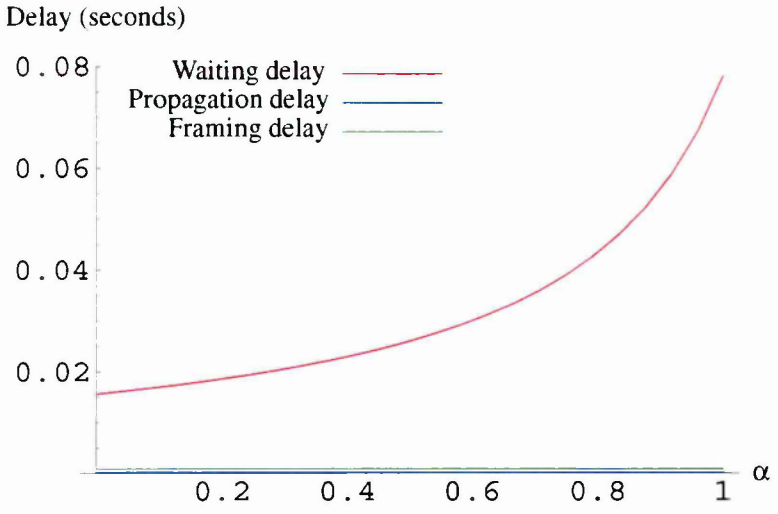


Figure 46 shows the aggregate data rate surface using a remote server with a mirror on a 10Mb/s local area network and upgrading the remote communication facility to 1.024Mb/s. Table 11 gives a summary of the model parameters

Table 11 Model parameters

Parameter	Value
d_r	8000
d_m	1
B_r	1024000
B_m	10000000

The affects of these changes are further increases in the aggregate data rate. As for the previous case, for values of $0 < \beta < 1$ the data rate is significantly higher when $\alpha \to 1$ than for $\alpha \to 0$. Furthermore the values of α and β for which the data rate of an individual remote user is better than that of an individual mirror user are much higher (Figure 47).

Figure 46 Aggregate data rate

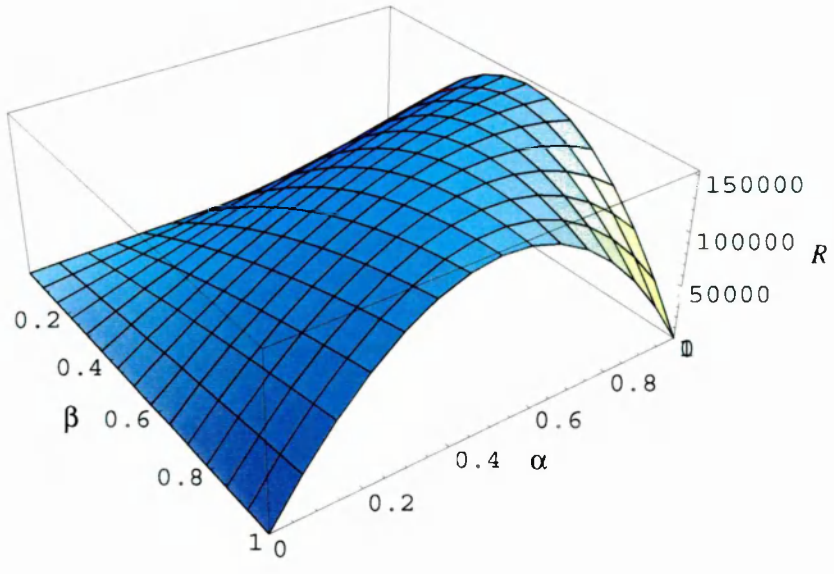


Figure 47 Data rate of individual remote and mirror users

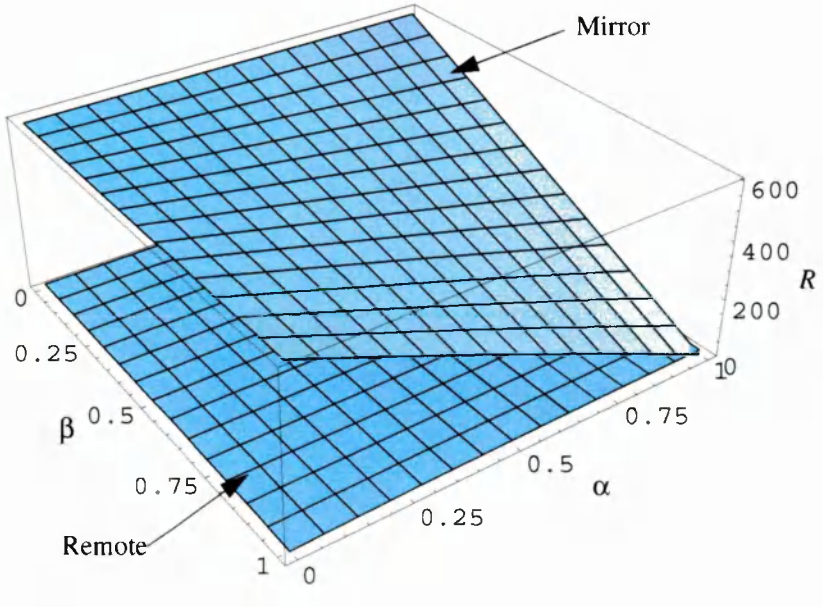


Figure 48 and Figure 49 show the remote and mirror component delays respectively.

Figure 48 Component delays for the remote

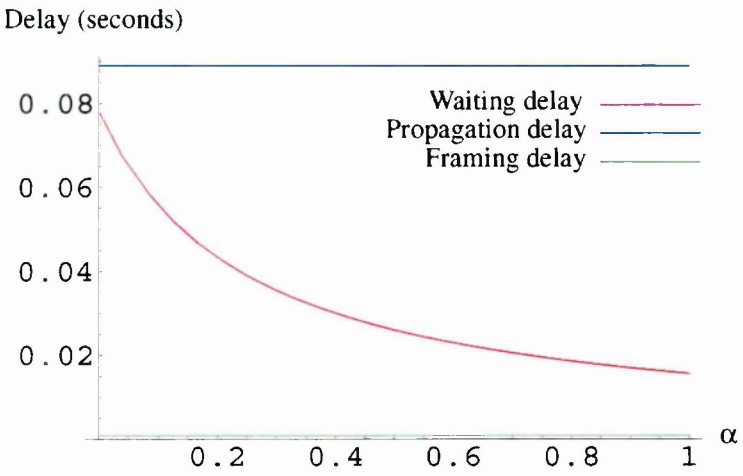
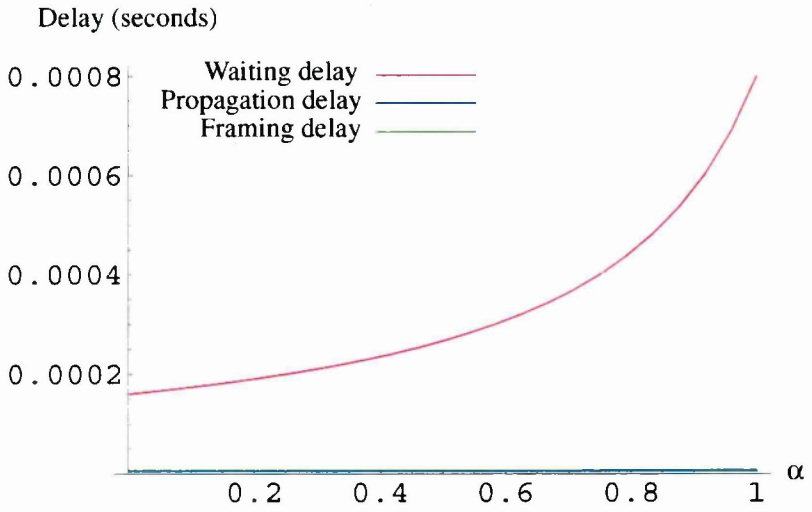


Figure 49 Component delays for mirror



What is significant about this network configuration is that, except for very low values of α, waiting delay is not the major contributor to the overall delay, rather propagation delay is. The overall delay to the mirror site is dominated by waiting delay, however delays are much lower than for the remote.

The final case examines a network configuration local area and wide network speed are equally high. Servers (remote and mirrors) and client area connect to an ATM network at 155Mb/s. Table 12 gives a summary of the model parameters

Table 12 Model parameters

Parameter	Value
d_r	8000
d_m	1
B_r	155000000
B_m	155000000

Figure 50 shows the component delays for the remote. The overall delays are entirely dominated by propagation delay. Both framing and waiting delay are relatively small such that the overall delay will be affected insignificantly by bandwidth upgrades.

While framing and waiting delays to the mirror (Figure 51) are the same as for the remote, propagation delays are much less (as expected). Again waiting delay was the highest contributor to the overall delay.

Figure 50 Component delays for the remote

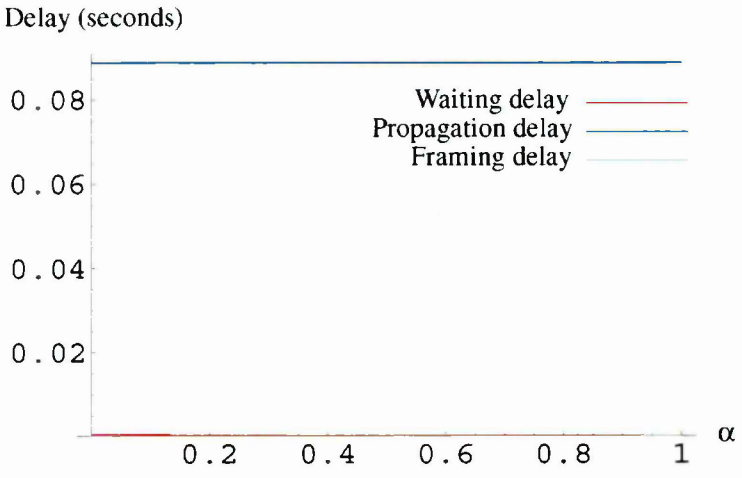
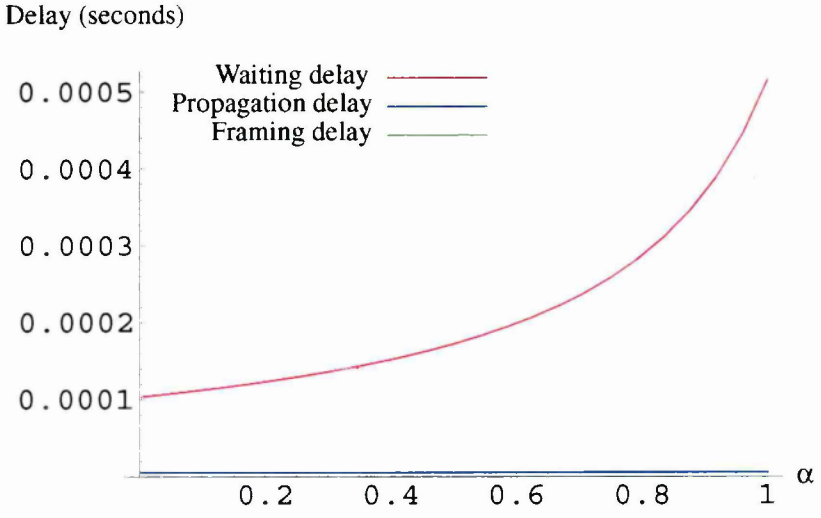


Figure 51 Component delays for the mirror



4.4 Load Balancing

Productivity gains can be made by moving the replicates closer to the user population, not only do users benefit from lower propagation delays, access speeds can be greater due to the reduced cost of shorter communication links.

The analysis above shows the values of α (the proportion of users using the cache) for which productivity is an optimum. But how can the network be controlled such that the traffic load is correctly balanced across remote site and mirror? One way is to assume that in attempting to achieve the best possible performance for themselves the user population would select between remote site and mirror in the correct proportions. However it would be naive to assume that they would do so. The only indication the user has that a particular server is congested (whether be the original or the mirror) is sense of slow response times. This is not necessarily enough for users make any judgement as to how to distribute their workload. They may not even be aware that the documents they want are replicated elsewhere in the network.

Technical solutions to this problem have been attempted. The NCSA used the "round-robin" facility of BIND[1] (Berkeley Internet Name Domain) [139] in order to load balance requests to their Web servers. Netscape [175] balanced the load across the machines of its Web site by implementing a randomisation scheme inside the (Netscape) Web browser. Requests to `home.netscape.com` resulted in a DNS lookup for `homeX.netscape.com`, where `X` is a random number between 1 and 32 (as there were 32 servers). This is fine provided the administrators of the Web site also develop the browser.

However, this solution only works when server and mirror(s) are co-located. Cisco offer a "Distributed Director" product as a means of selecting mirror sites that are not co-located [49]

1. BIND is the mostly widely used DNS internet server

> if a DNS query for a named service, such as www.cisco.com, is sent from a user in Los Angeles, California, the DistributedDirector would respond with the IP address of the Web site in San Jose, California. But if the query came from a user in Sweden, that user would be directed to the Cisco Web site in Brussels, Belgium.

While directing requests to the nearest mirror reduces propagation delays the analysis in this chapter shows that the overall delay can be high if the server (or mirror) is busy, which is why the manufacturer promises future enhancements whereby this product will have the "ability to factor in servers' load conditions in addition to geographic proximity" [49].

However users relinquish control over mirror selection when technical solutions are "imposed", stale data will occasionally cause people to have to rework tasks. This repetition reduces the overall productivity.

4.5 Stale Data

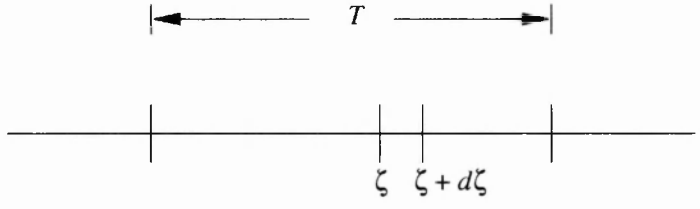
Documents on Web servers will be added, deleted and changed. These changes have to made on the respective mirrors. In order to take advantage of temporal traffic dispersion updates to mirror sites are typically delayed until off-peak periods. This means however that for a period of time the mirrors are serving *stale* data. Users' may measure productivity, not only in terms of data rates, but also in terms of its currency. Baentsch et. al. [12] outline the problem from the users point of view of automated mirror selection

> administrators, for example, always strive to optimize network utilization, while users want to obtain the most up-to-date version of any data.

A way of overcoming stale data is to increase the frequency of updates or do triggered updates. However this causes increases in traffic volumes. The following analysis shows the likelihood of accessing stale data as a function of document change and update frequency.

The probability of the first revision being made in the interval between a time ζ after the beginning of an update interval and the time $(\zeta,\zeta + d\zeta)$ is the probability of having no revisions earlier multiplied by the probability of any revision in the interval (Figure 52).

Figure 52 First document change in the interval T



Assuming an exponentially distributed creation rate, if k is the expected rate of creating revisions then the probability of no revision before time ζ, that is in an interval ζ, is

$$e^{-k\zeta} \tag{51}$$

The probability of at least one revision in the interval ζ and $(\zeta,\zeta + d\zeta)$ is one minus the probability of no changes in an interval $d\zeta$

$$1 - e^{-kd\zeta} \tag{52}$$

The expression in (52) can be expanded as a series

$$kd\zeta - \frac{k^2 d\zeta^2}{2} + \frac{k^3 d\zeta^3}{6} - \frac{k^4 d\zeta^4}{24} + \frac{k^5 d\zeta^5}{120} \ldots \tag{53}$$

Since $d\zeta$ is small, high order terms in $d\zeta$ can be ignored. The probability of no change taking place before the interval multiplied by the probability of at least one change taking place in the interval is

$$e^{-k\zeta} \cdot kd\zeta \tag{54}$$

Suppose, now that documents are retrieved according to an exponential distribution with a parameter of λ. Stale documents will be received from the mirror if the document has been revised but the mirror has not yet been updated. That is stale documents are those requested from the mirror at a time ζ after the previous mirror update when the revision is presumed to take place and before the next update at a time T after the previous update. The expected number of accesses in the interval (ζ, T) is

$$\lambda(T-\zeta) \tag{55}$$

Thus the expected number of stale documents retrieved in an interval between updates is given by the aggregate of the expected number of documents retrieved in the interval ζ to T times the probability of a change taking place at time ζ

$$\int_0^T \lambda(T-\zeta)ke^{-k\zeta}d\zeta \tag{56}$$

Integrating (56) gives the expected total number of stale documents retrieved

$$\frac{\lambda}{k}(e^{-kT}-1+kT) \tag{57}$$

The expected number of documents retrieved in the interval 0 to T is λT hence the expected proportion of documents that are stale is

$$\frac{\lambda}{k}\frac{(e^{-kT}-1+kT)}{\lambda T} \tag{58}$$

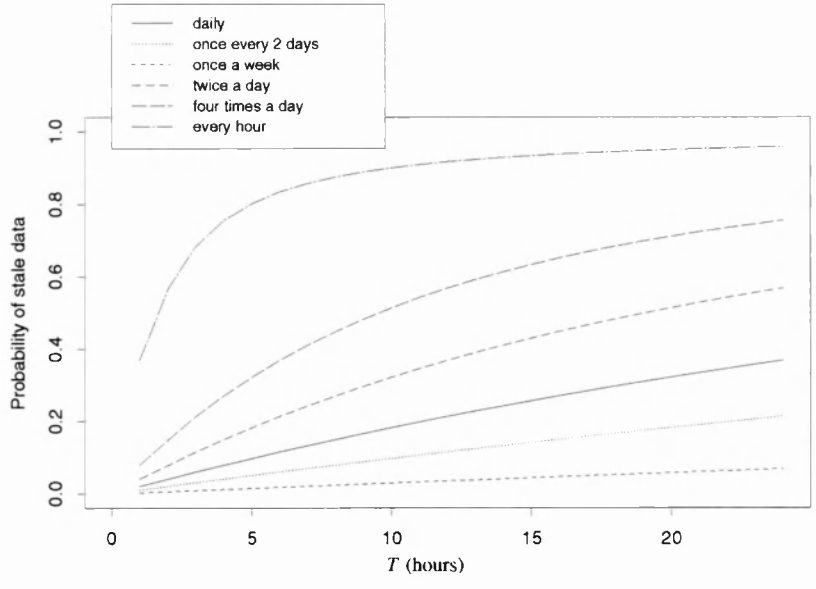
Simplifying this expression gives

$$\frac{e^{-kT}-1+kT}{kT} \tag{59}$$

The graph in Figure 53 shows the probability of accessing stale a document for documents that on average changes, once a week, once every 2 days, daily, twice a day, four times a day and every hour. It can be seen that in order to keep the proportion of stale

documents low that the update frequency must be high (that is T is kept low relative to the revision rate K).

Figure 53 Probability of stale data



Decisions about the distribution of mirrors and the updating policy of mirrors is influenced by factors such as document sizes, available bandwidths and the rate at which documents are altered. The values of these factors depend on the particular situation and circumstances and it would be hard to create a generalised policy. Improvements in performance can be brought about by considered choices but only with details of the specific context.

4.6 Replication Experiment

An experiment was performed where replication was used to bring about performance improvements in using an application called Teamwork in distributed wide area network environment. It was carried in order illustrate how bringing data closer to the user can reduce access delays and network traffic volumes.

Teamwork is a software engineering package that was used by software developers at Lucent Technologies UK (then AT&T) and the Bell Laboratories in the US. It was common for developers at both sites to want to access and modify the same data. If a developer needed to work on a data model located on a server at a remote site, the file containing the model had to be transferred over the network (using UUCP, RCP or FTP) to a local server. The model was then shipped back when the developer had finished modifying it. This meant during the time the developer was working on the data model there were actually two copies of it. This scenario is known as the readers/writers problem [230]. While one developer was working on the copy of the data model, another developer could modify the original. When the (now modified) copy was shipped back to the remote server the modifications made to the original would be over written.

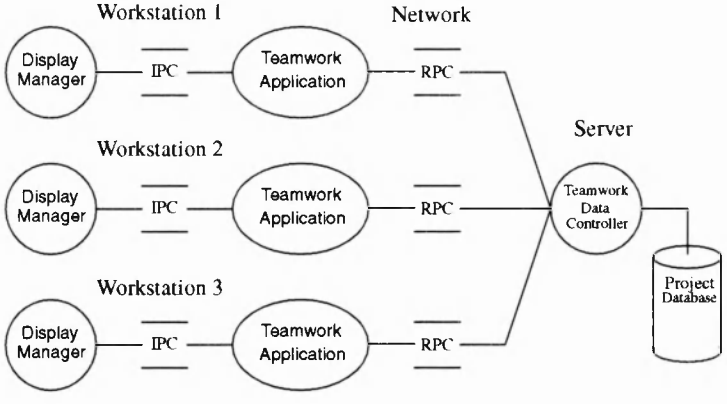
The Teamwork architecture consists of a Display Manager (DM), a Data Controller (DC) server, a Project Database and a number of interactive applications (see Figure 54). The DM process executes on a user's workstation and provides the Graphical User Interface to the interactive Teamwork applications.

There are a number of interactive Teamwork applications. Examples of which include graphical editors for creating and modifying models for analysis and design. In the workstation/server environment both the DM process and the applications processes execute on the user's workstation through an interprocess communication (IPC) mechanism.

The DC server process controls access to the Teamwork Project Database. An application process requiring a data model from the Project Database makes a request to the DC server through a Remote Procedure Call (RPC). If the required data model is not already in use, it is copied to a file in a server directory that is NFS mounted on the user workstation. The application process then accesses the file containing the data model

through NFS. The DC server provides *concurrent transparent* access to the Project Database within the distributed environment.

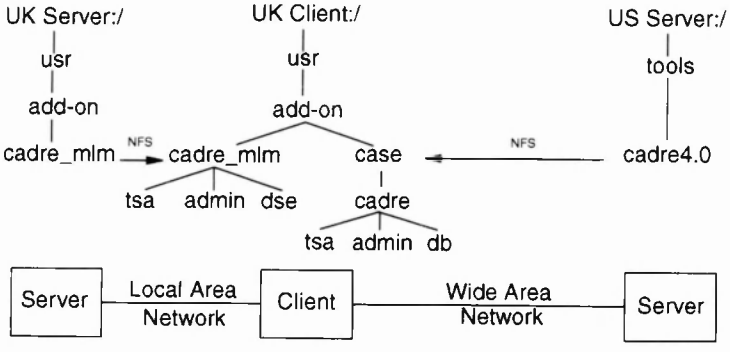
Figure 54 Teamwork architecture



Typically workstations in the UK mounted file systems for Teamwork from a server in the UK. However by NFS mounting the Teamwork directory over the wide area network from a US server, US data models could be accessed in a controlled way (via the DC server process) thereby eliminating the readers and writers problem.

A trial was set up where one workstation in the UK was configured to use the Teamwork server in the US (Figure 55) and during Teamwork sessions, the network usage was monitored.

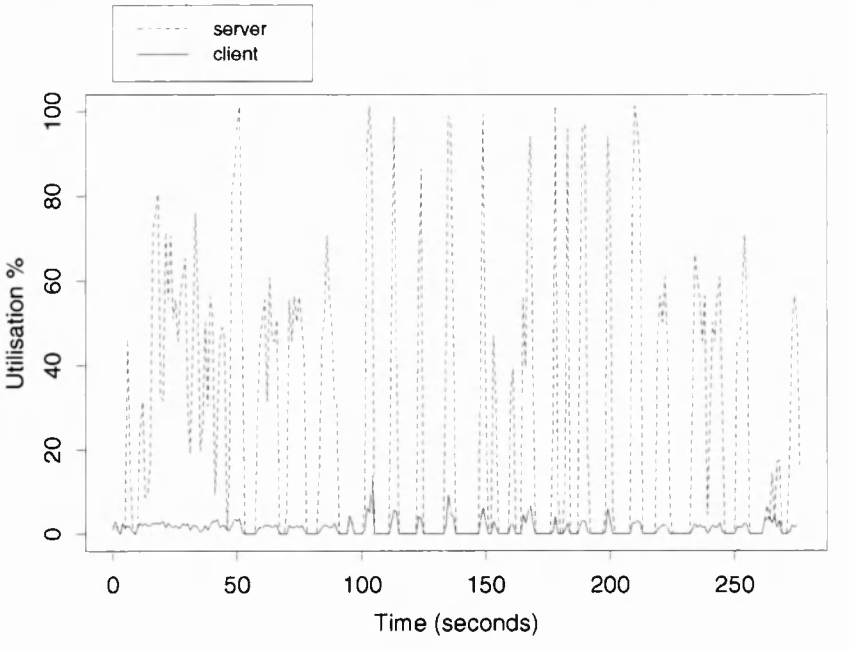
Figure 55 File systems remotely mounted on a UK client



The start of the Teamwork session was found to be quite bandwidth intensive, more so than for any other part of the session. This high utilisation of the wide area network serial links was caused by the initial start-up procedure. Typical of many client-server type distributed systems, the network usage was asymmetric, in that the load in one direction was greater than in the opposite direction. In this case the server-to-client direction placed the highest load on the network. Load in the client-to-server direction was small in comparison and represented the client RPC requests.

The graph in Figure 56 shows the percentage utilisation of the wide area network between the UK and the US. The traffic was very bursty with periodic peaks briefly consuming the entire bandwidth of the serial link. The mean load throughput (server-to -client) was 57kb/s for the duration of the start-up procedure. Consequently, the start-up procedure took nearly five minutes.

Figure 56 Teamwork start-up: application executables from US server



It is interesting that the traffic bursts appeared to be at regular intervals. This gave some insight as to the type of data that was being transferred over the network. The periodic bursts of high load were due to the Unix demand paging mechanism. When an item of data is not in main memory, the operating system *faults* in the appropriate page from disk or swap. In this case the executable image resided on the disk of the server, so text pages had to be *paged in* over the network. Unix does not require that the whole executable image of an application process be resident in main memory in order to execute it. It only needs the page containing the instructions it is currently executing. Therefore, the text pages are paged in *on demand.*

Closer examination of the network traces showed the type of data being transferred between server and client to be:

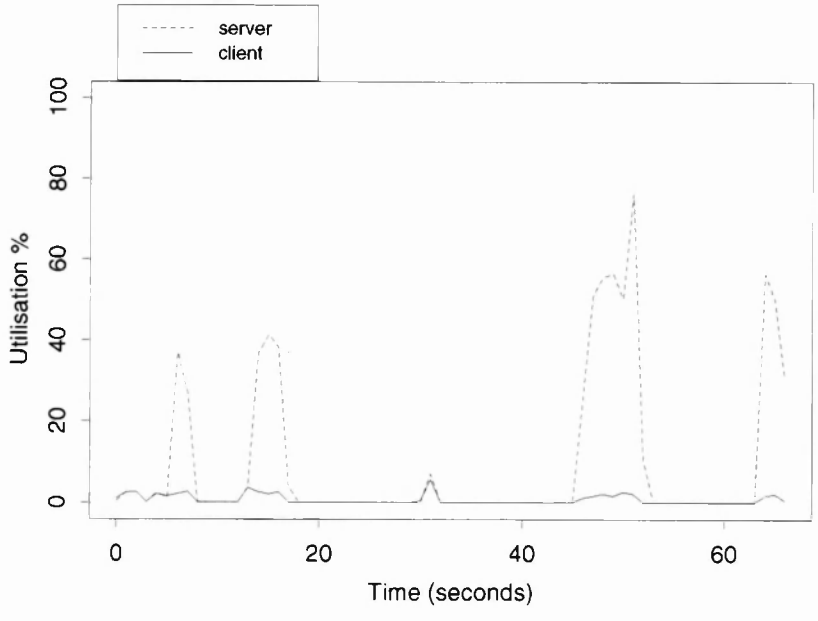
- binary images
- object libraries
- log file data

- configuration file data

It was found that the majority of data transferred was the Teamwork application executables (binary images and object libraries). As the Teamwork application executables are mirrored on a server in the UK and are infrequently modified, there is no reason to access them from the US over the wide area network The client workstations were reconfigured to pick up executables from the local UK server, thus confining demand page activity of executables to the local area network. The DC server and model data were still accessed from the US.

The graph in Figure 57 shows the utilisation of the wide area network during the Teamwork start-up procedure using the mirrored executables and object libraries in the UK. The time to complete the start-up procedure was reduce from five minutes to just over a minute. Furthermore there mean traffic load for this period was reduced (from 57kb/s) to 19.2kb/s.

Figure 57 Teamwork start-up: application executables from on-site server



From the users point of view the application was quite usable. The were some noticable delays while data models were shipped from the server's disk system into workstation memory, but they were not unacceptably long. Highly interactive parts of the Teamwork session was confined to the local workstation so there was no difference in performance compared to that of the local site environment.

Though it was necessary to have access to data models resident on a US server, it was very inefficient to fetch the executables from the US server when the same executables were *replicated* on the UK server. It can be seen from the Teamwork example that it was not necessary to use the US server for all "data"

4.7 Chapter Summary

In the chapter the packet delay model presented in CHAPTER 2 was used to develop a productivity model for accessing Web documents from mirror sites. Using mirror sites has a number of benefits:

- Lower propagation delays are incurred if a document is fetched from a mirror site that is in closer proximity to the user than the original remote site (or any other mirror site).
- Fetching documents from a mirror diverts traffic away from a (potentially) busy remote server and a congested part of the network.
- The cost of communication facilities increases with distance. Therefore it is financially viable to have higher speed links to (closer) mirror sites than the original remote site.

Typically the performance benefits are best when the mirror used is closest to the user, but when communication facility speeds are low (in the order of kb/s) these benefits are small as overall delays are dominated by packets waiting in buffer queues. Propagation delays do become significant when bandwidths are high (in the order of Mb/s), this is

when (closer) mirror sites yield higher productivity gains for those using them than users of (distant) remote servers.

However, the results of this analysis show that close proximity does not necessarily ensure the good performance. Mirror sites distributed throughout the globe will exist in different time zones. Given that user accesses exhibit a temporal distribution [6] it reasonable to assume users will share the same network peak hours with *nearby* mirrors. For peak time users in one part of the world, it is likely that distance mirrors will be operating during network quiet periods. Despite the higher propagation delays the absence of congestion may make a distant mirror (or the original remote server itself) may provide a user with higher productivity (particularly when bandwidths are low and waiting delays tend to dominate).

What is interesting about this analysis is that regardless of the difference in bandwidth and distance the optimum productivity does not deviate much from the point where users are load balanced equally between remote and mirror.

So at the point of optimum productivity there is a near equal split between one set of users experiencing low round trip delays (mirror users) and another set experiencing longer delays (remote users). In which case there is a dilemma between achieving optimum productivity from the network and creating two classes of user in terms of the performance they experience individually.

The experiment with the Teamwork application showed the performance benefits of replicating data when accessing data from a remote server (in this case in the US). During a Teamwork session binary images, object libraries, log files and configuration files are fetched from the server. The transfer of the binary images and object libraries (executable code for the applications) over the wide area network constituted the majority of traffic generated by the Teamwork application. Executable data is read-only and are

therefore not subject to the readers/writers problem, so replicating them on a local server in the UK the amount of traffic generated (over the wide area network) by the initial start-up procedure was considerably reduced, as was the time to complete.

This experiment also illustrated how understanding the data access characteristics of the Teamwork application influenced the choice of data to replicate. Not all the data could be replicated because the log files and configuration files required write access. Replication of this data in the UK, (while bringing about further performance improvements) would have resulted in a situation similar to the problem of stale data with mirror (the difference being that data also changes and the "replicating" server not just one the remote).

Mosedale et. al. [175] report that "wholesale mirroring of Web pages" is generally deemed "not an effective way to reduce latency on the networks". A user access model based on "site popularity" may not be appropriate because, according to Baentsch et. al. [11] "only a small subset of documents offered by any given server that are very popular".

CHAPTER 5 *Caches*

Caches replicate data on a per document basis, whereas mirrors replicate entire sites. Furthermore, while a mirrors replicate data from a single *fixed* site, caches can replicate documents from different sites. Caches have a storage capacity limit so not all documents on the Web (of which there are many) can be "brought closer to the user".

Caches must be selective about which documents they replicate and which documents they do not. Cache occupancy increases with each cache miss. When a cache reaches its storage capacity limit it has to purge some of its documents, otherwise it is unable to replicate any further requests. This process is called *Garbage Collection*.

Web documents will vary in size, distance from the receiver and frequency of access. Consequently so too will the benefits of caching some documents in favour of others. The decision as to which documents to purge is called the *Cache Replacement Policy*.

Effectively the replacement policy determines the propagation delay incurred in accessing a document, that is documents that can be accessed from the cache will incur propagation delays less than those that have to be fetched from the remote server (not in the cache).

There are also performance issues that need to be considered regarding garbage collection. Over zealous purging of documents from the cache will result in more frequent requests to remote servers (because certain popular documents are no longer replicated). This will mean higher propagation delays and increased traffic on long haul communication facilities. However if the cache is conservative about purging documents it will tend to reach its capacity limit more rapidly resulting in frequent garbage collection. The cache itself could then become a bottleneck, spending a high proportion of its time doing garbage collection rather than serving documents.

Garbage collection and the replacement policies adopted by the cache has therefore, a direct impact upon network performance. Ideally these policies should maintain high cache occupancies and low garbage collection frequencies.

Research into the performance benefits of Web caches has been quite extensive (see for example, [6], [10], [11], [12], [157], [177] and [218]). Rather than the performance benefits brought about by the cache this chapter looks at the performance of the cache itself. The effect of different policies on cache hits rates and garbage collection frequencies are examined using trace simulations.

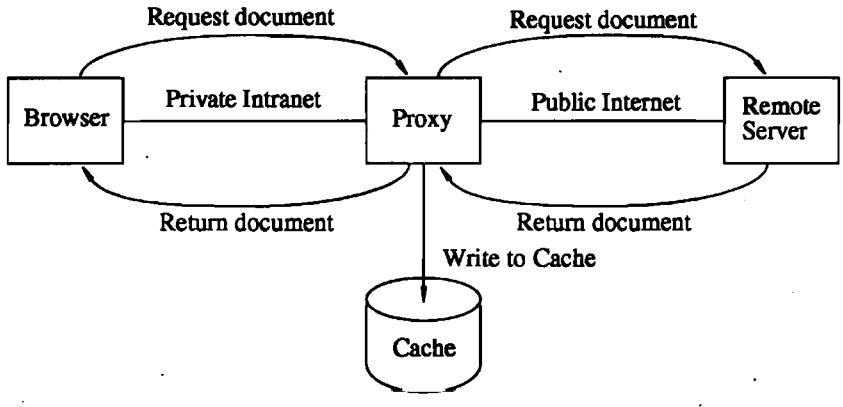
Also this chapter sets out to show that insights into caching policies can be gained from analysing the growth in cache occupancy. Cache references form a sequence of hits and misses. By making an assumption that cache references occur as independent Bernoulli trials an analytical model of cache growth can be constructed using the Binomial distribution. What is good about this model is that it is fairly straight forward to construct. However, as with a number of stochastic processes in communication systems, the assumption of independence is often an invalid one (see for example [22], [62] and [76]). Predictions made by models that assume independence can vary considerably from actual network environments where stochastic processes exhibit a high degree of correlation.

Using data from the log files of a Lucent Technologies proxy server the assumption of independence in cache references is tested. A second cache growth model is presented using Monte Carlo simulation techniques based upon correlated references and, along with the Bernoulli model is compared with cache growth measurements from actual cache data.

5.1 Cache Operation

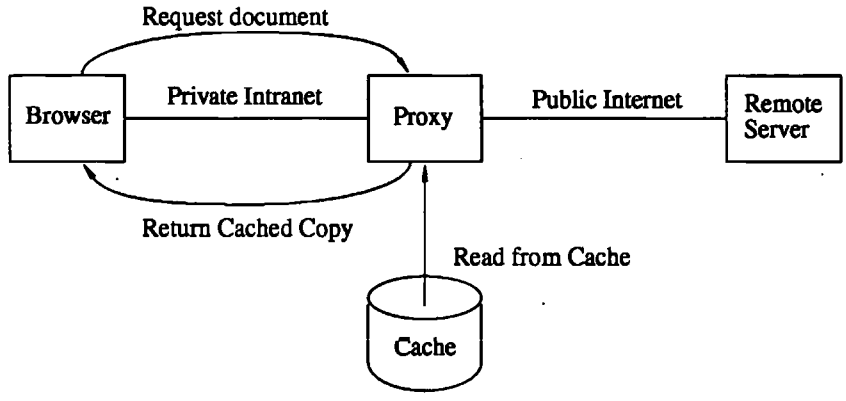
Caching can be implemented on either the browser or a proxy server (or both). The analysis performed in this section focuses on a proxy server cache. A proxy cache writes any documents it has fetched to disk before forwarding the documents on to the requesting browser. The proxy will return the copy in the cache for any subsequent requests for the same document. A cache miss (Figure 58) occurs when the proxy does not have a copy of the document in its cache and has to fetch it from the remote server.

Figure 58 Cache miss



Subsequent requests for the document will be served from the cache. Figure 59 shows a cache hit.

Figure 59 Cache hit



Typically proxy servers within the Lucent Technologies organisation run on firewalls. However Lucent Technologies located in Malmesbury operates a proxy server despite not being a firewall site, Hence, using the proxy merely for its caching functionality.

Cache memory is typically smaller than the memory space for which it caches. So at any one time the cache can only store some fraction of the contents of the whole memory space (at any particular point in time the data in the cache is called the *working set* [4]).

Typically data requests exhibit *locality of reference*. Locality of reference is the phenomenon that recently accessed data is likely to be accessed again in the near future. So any subsequent references for an item of data can be serviced by the cache.

The effectiveness of the cache depends upon how frequently data items are accessed and can be quantified by the cache *hit rate*

$$\frac{\text{number of cache hits}}{\text{number of references}} \tag{60}$$

where a high hit rate is desirable. That is, users will generally experience lower propagation delays as more documents are served from the cache and less from the remote server. Drawing upon the analysis of mirror performance, if d_r is the distance to the

remote server from the requesting browser, and d_c is the distance to the cache then the mean propagation delay $\overline{D_P}$ is

$$\overline{D_P} = \frac{P_m d_r + P_h d_c}{c_F} \quad (61)$$

where P_m is the probability of a cache miss and P_h is the probability of a cache hit. ($P_m + P_h = 1$). Speed of light through the transmission medium is c_F. Provided $d_c < d_r$, $\overline{D_P}$ will decrease as P_h increases (and consequently P_m decreases).

An analysis of the Malmesbury proxy log files between 27th February 1995 and 29th December 1996 was carried out. Figure 60 and Figure 61 show the results of this analysis for the Requests Serviced and the Bytes Transferred respectively. These statistics are broken down three ways:

- The "proxy to client" curves show the total number of requests made to the proxy and the resultant number of bytes transferred to the Web browsers.
- The "remote to proxy" curves show the number of request that caused a cache miss and the amount of wide network traffic that was generated as a result.
- The "cache to client" curves show the number of requests that were found in the proxy's cache and the amount of network traffic that was *avoided* as a result of having a proxy cache.

Figure 60 Requests serviced

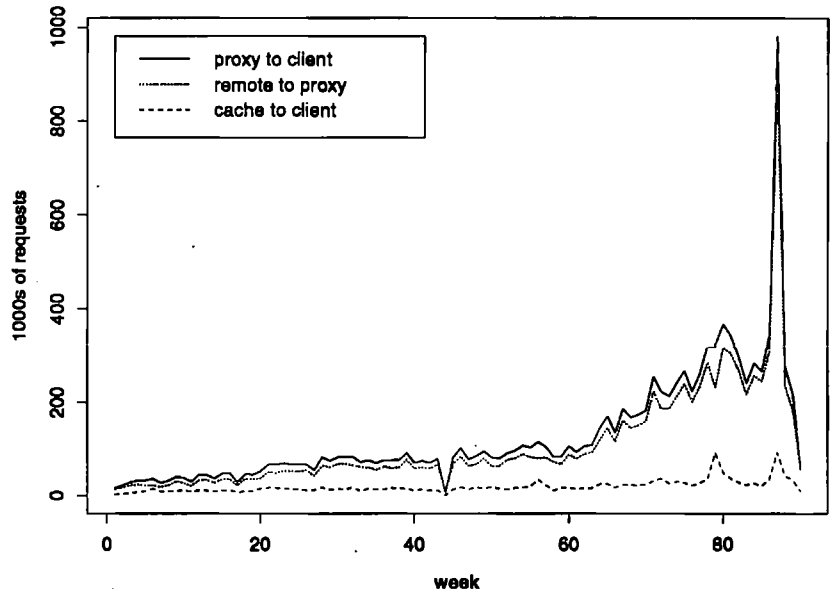


Figure 61 Bytes transferred

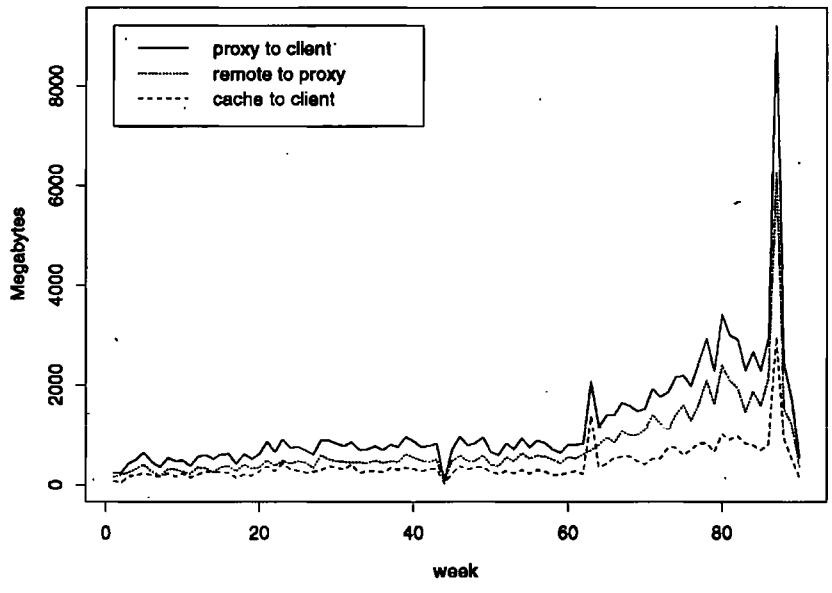
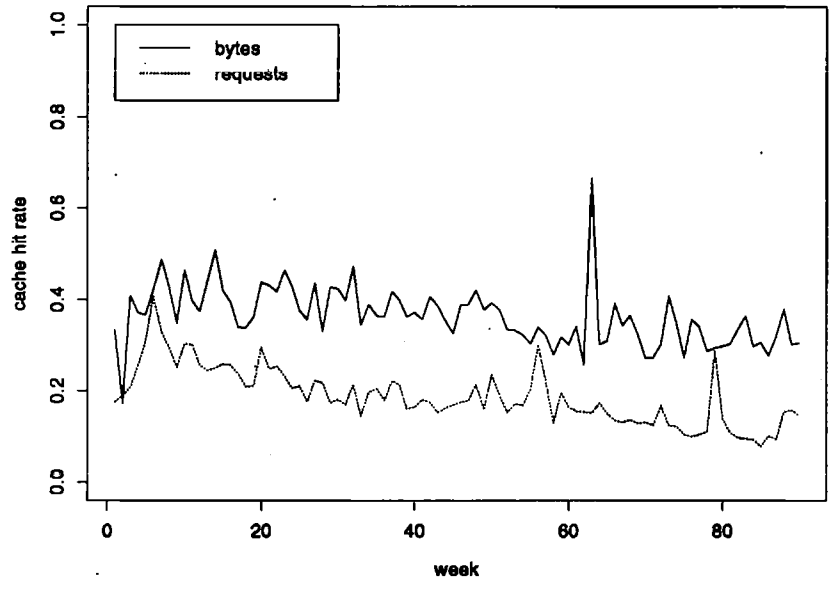


Figure 62 shows the cache hit rates for Requests and Bytes. Over this period the average Request hit rate was 15% and the average byte hits rate was 35%. This is quite low and

consistent with the 20% hit rate reported by Muntz and Honeyman [177] (it is not specified whether this figure is for request or bytes). Arlitt and Williamson [6] attribute low cache hit rates to the lack of temporal locality in Web references. However Danzig [64] and Smith [218] report hit rates as high as 50% and 55% respectively.

Figure 62 Cache hit rate



Danzig attributes high hit rates to the use of cache hierarchies, but Muntz and Honeyman [177] claim that hierarchies diminish locality of reference and just become "latency *creation* mechanisms". Baentsch et. al. support Danzig's view on the use of hierarchies and refute the significance of the delays introduced by caches in wide area networks

> In the wide area network environment of the Internet, latencies induced by caches are negligible [12].

However Smith relates the performance issues of the UK national Web cache where

> the demand has been so great that queues on the server resulted in using the cache actually being slower than going direct [218].

5.1.1 Replacement Policies and Garbage Collection

Caches are limited in size and are usually many times smaller than the data space they cache for. At some point a cache will reach its capacity, making it necessary to flush all or part of it.

The choice of which files to clear out constitutes the *replacement policy*. Baentsch et al. [12] present the following examples of cache replacement policies:

- Random. Documents are cleared out at random.
- Least Recently Used (LRU). Documents are cleared out on the basis of being least recently referenced.
- Least Frequently Used (LFU). Documents are cleared out based upon their (un)popularity.
- Time To Live (TTL). Documents are cleared out if they have expired or are close to expiry.
- Least Caching Profit. Documents are cleared out based on having the least amount of benefit with respect to loading latency.
- Never clear out the cache.

The Netscape proxy server schedules a Garbage Collection process when the cache reaches its limit to clear out some documents. Garbage Collection can also be scheduled periodically (e.g. once a day, outside of prime time) in order to clear out expired resources.

The object of a cache replacement policy is to evict documents that will not be needed soon. Clearing out a document that is requested in the near future (a *capacity* miss) is clearly undesirable. A replacement policy where the cache is never cleared out may

appear at first to be unfeasible, requiring a cache of unlimited size. Baentsch et. al. [11] report low document hit rates and byte hit rates, 21.3 percent and 16. percent respectively. This is consistent with the low cache hit rates for Lucent Technologies proxy server at Malmesbury (document hit rate 15 percent, byte hit rate 35 percent as stated above). Analysis by Baentsch et al. [11] claims that document hit rates of 56.5 percent and byte hit rates of 40.6 percent are theoretically achievable with an unlimited cache (no garbage collection).

The cache size of the Malmesbury proxy server has increased from 100 Megabyte to 6 Gigabytes in less than four years (a 60 fold increase) which has been facilitated by the reduced cost of disk capacity. It may therefore be feasible to merely grow the cache as it approaches its capacity. Using WORM (write once read many times) devices would increase the cache capacity further but there is still a need for recognising stale files.

5.1.2 Cache Coherence

Stale data is an inherent problem with replication methods. Proxy servers try to ensure cache coherence by issuing an HTTP conditional GET request to the remote server, for example:

```
GET http:/www.lucent.com/ HTTP/1.0
If-Modified-Since: Thu, 04 May 1995 11:04:23 GMT
```

The server (www.lucent.com) would respond in one of the following ways:

- If the document is inaccessible the server returns a message with a status code of `4XX` back to the proxy.
- If the document no longer exists, the server returns a message with the status code of `404` (Not Found) back to the proxy.
- If the document is accessible, and the `If-Modified-Since:` date. is less than the document's `Last-Modified:` date, the remote server returns a mes-

sage to the proxy with a status code of 304. In this case no resource body is returned, and the proxy server returns the cache copy.

- If the document is accessible, and the `If-Modified-Since:` date. is not less than the resource's `Last-Modified:` date, the remote server returns the resource body with a status code of 200 in the message header.

Using the conditional GET request allows the proxy to serve a resource (that has not been modified) from its cache (thus reducing latency) and to guarantee coherence. However, guaranteed cache coherence requires interaction with the remote server which introduces an inevitable latency.

The proxy server can make a *heuristic* assessment of whether a cached copy of a resource has expired (and become inconsistent with the original) without contacting the remote server. There are a number of ways it can do this:

- Use the HTTP `Expires:` parameter in the resource header.
- Use the HTTP header `Last-Modified:` parameter in the resource header.

Using the `Expires:` parameter is the most accurate method of assessing whether a resource has been modified without contacting the remote server. The `Expires:` parameter explicitly specifies the date on which the cache copy of the resource will expire. If the current date is greater then the date specified by the `Expires:` parameter then the proxy checks the remote server (of course, the resource on the remote server may not have actually, changed despite the current date exceeding the `Expires:` date, so the proxy still does a conditional GET).

The `Expires:` parameter, however, is not mandatory. Absence of an `Expires:` parameter causes the proxy server to use the `Last-Modified:` parameter. The proxy server estimates that a resource's time to expire as a proportion of the time between the current date and the resource `Last-Modified:` date. The time between the current

date and the resource `Last-Modified:` date multiplied by a factor that is configurable by the proxy administrator.

The smaller the factor the more cautious the proxy is about checking the remote server. Large values will reduce latency but will increase the risk of returning stale data from the cache.

The philosophy of the "`Last-Modified:`" heuristic is that: resources that have not been modified for long period of time are not likely to change in the near future. The converse of this philosophy is that resources with a recent `Last-Modified:` date *are* likely to change in the near future.

In addition to the heuristic approaches described above, the Netscape proxy offers a third method of reducing interaction with the remote server (and latency). A time period can be specified whereby the proxy server will not do an expiry check to the remote server if this time period has not elapsed since a resource was last referenced. As with the `Last-Modified:` heuristic, large values reduce latency but are less cautious about serving stale data.

Turning the `CacheExpiryCheck` to `Off`, relies on periodic execution (typically daily) of the Garbage Collector to purge the cache of stale resources. This is by no means an ideal method of cache coherency maintenance, and it may not necessarily reduce access latency. Indeed it may even cause it to increase due to excessive use of the "Reload" operation by users who have no confidence that resources returned by the proxy are up-to-date.

5.1.3 Prefetching

Typically documents are cached on demand, that is when they are first requested. Prefetching attempts to reduce access latencies by doing *lookahead* retrieval of Web documents and caching them prior to their request. Prefetching can be implemented in

the browser or the proxy, but the obvious advantage of proxy prefetching is that it serves multiple users. The effectiveness of prefetching is dependent upon the performance of the prefetch policy, that is the successful anticipation of future requests.

Prefetching is not exclusive to the Web and is used in other cache systems. The Unix buffer cache is one example [9]. Unix organises data in mass storage (disks) as files. Each file is a collection of disk blocks that can be accessed independently. So if a process requires an item of data from a file, only the disk block containing that data is loaded into main memory rather than the whole file. However, if a process wants all the data in a files then the file must be loaded into memory using multiple I/O (Input/Output) requests for successive disk blocks.

So if the file system receives a request for a disk block it has no way of knowing whether it is merely a one off request for a single item of data or the first of many consecutive requests. The Unix buffer cache system implements a disk block read-ahead algorithm. A request for a disk block results in that block *and* the next being fetched from disk (after checking the cache).

It is difficult to predict in advance what Web pages users are going to access. Wang and Crowcroft report [242] interdependencies between Web references. Indeed the very nature of the World-Wide Web is that information is organised as a "web" of pages interconnected by *hyperlinks*. However Wang and Crowcroft [242] argue that extra traffic due to aggressive prefetching can actually increase access latency due it additional bandwidth consumption. Furthermore incorrect prefetches increases the occupancy of the cache which could lead to unnecessary garbage collection.

Prefetching needs to be selective but it would require quite sophisticated (and rapid) algorithms to predict, with a high success rate, users' future references. Analysis by

Wang and Crowcroft [242] shows that prefetching is only effective when either traffic is light or prefetching efficiency is high.

The benefits of *batch* prefetching are not so dependent upon prefetch efficiency. Batch prefetching downloads Web documents during off peak periods, when network traffic is light and the penalties for incorrect prefetches are less. Prefetching entire sites is equivalent to mirroring and could consume large amounts of disk space. Furthermore, according to Baentsch et al.

> reserving a considerable amount of cache space exclusively for replication decreased overall performance [11].

Therefore the prefetch policy still has to be selective. For example documents that change periodically, like weather forecasts, could be prefetched. Such a prefetch policy also reduces problems of cache coherence as frequently changing documents are kept up-to-date.

5.2 Cache Growth

The assessment of garbage collection and replacement policies relies on accurate models of requests and document availability. As with any model accuracy must often be traded with computational simplicity.

The first model predicts cache occupancy from the Binomial distribution by assuming that cache references occur as independent Bernoulli trials. However analysis of a proxy server's log files reveals that cache references are not independent. So a second model was developed using Fractional Autoregressive-Integrated Moving Average processes and Monte Carlo simulation techniques. The accuracy of these two models is verified against data from a proxy server's log files.

5.2.1 Analytical Model

Cache references result in sequences of hits and misses and can be modelled by stochastic processes where each trial results in one of only two possible outcomes. By assuming that cache hit/miss sequences occur as Bernoulli trials, the Binomial distribution can be used to predict the number of cache misses experienced in a given number of requests to the cache. From this, increases in cache occupancy can be determined.

The Binomial distribution gives the probability $P(Y = i)$ of exactly i events in n number of trials

$$P(Y = i) = \frac{n!}{i!(n-i)!} \cdot p^i (1-p)^{(n-i)} \tag{62}$$

If the proxy server receives n document requests per day, then the probability that exactly i requests resulted in a cache miss (and thus increasing the cache occupancy by i files) can be calculated. The probability that the cache occupancy increases by i files or more (in n requests) is

$$P(Y \geq i) = \sum_{k=i}^{k=n} P(Y = k) = \sum_{k=i}^{k=n} \frac{n!}{k!(n-k)!} \cdot p^k (1-p)^{(n-k)} \tag{63}$$

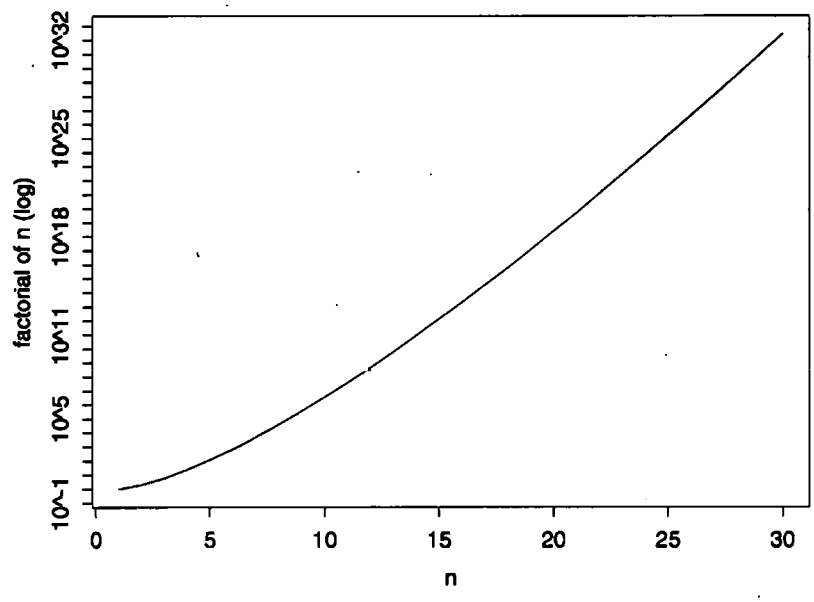
where p is the probability of a cache miss. For values of i which equal or exceed the amount of files cleared out earlier by the cache (63) gives the probability that the cache will reach its capacity.

However, typical values for n and i could be anything between 10^4 and 10^6 which would make values of $n!$ and $i!$ astronomically high (Figure 63 shows the growth of factorials). Computing such high numbers and doing arithmetic with them presents a problem. Knuth estimated that 10! factorial would take about an hour using early 1970s computer technology and that

> ...the number 10! represents an approximate dividing line between things which are practical to compute and thing which are not [135].

Fortunately late 1990s computer technology can calculate 10! somewhat faster (less than 2.5 microseconds on a 147MHz Sun SPARC Ultra). However with machine precision arithmetic there are other limitations. Using a 32 bit computer, 12 is largest factorial that can be stored as $12! < 2^{32} < 13!$.

Figure 63 Growth of factorials



Factorials up to 7,000 were computed using the arbitrary precision arithmetic routines in [198]. The times to compute these factorials are shown in Figure 64. It can be seen that computation times increase exponentially, with 7000! taking over 32 hours.

The Mathematica command `Timing[n!;]` was used to derive the CPU times to compute large factorials (10,000 to 100,000) on a Sun SPARC Ultra I. The results are shown in Figure 65.

Figure 64 Time to compute factorials using arbitrary precision routines

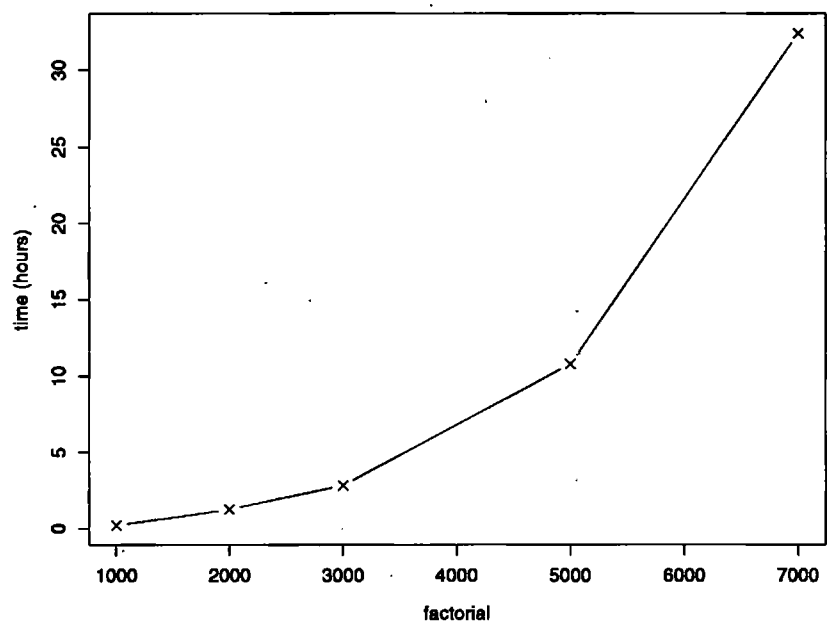
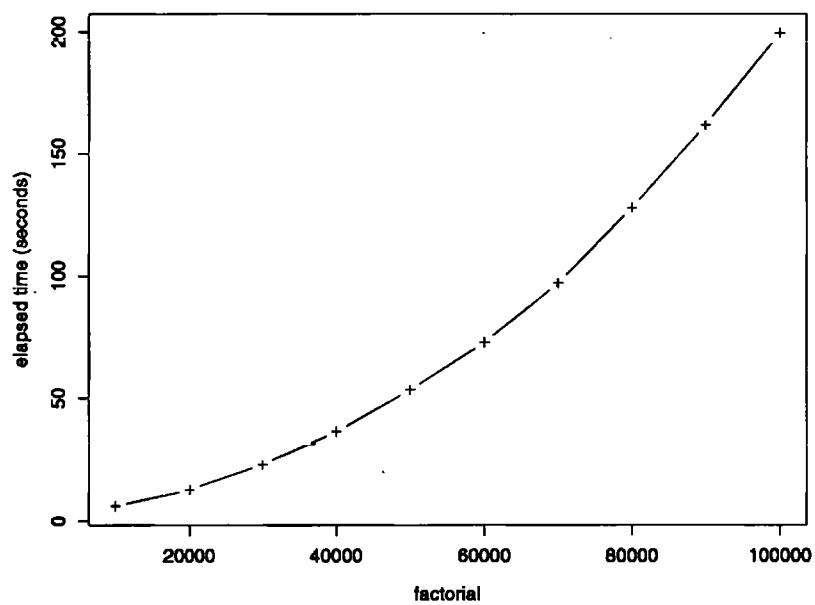


Figure 65 Time to compute factorials using Mathematica



Furthermore the Mathematica command `Binomial[n,i]` provides a means of deriving $\binom{n}{i} = \frac{n!}{i!(n-i)!}$. However, the time to compute $\binom{10000}{i}$, for $i = 1, 2..., 10000$ was 21.67 minutes!

While packages like Mathematica make dealing with expressions that involve large factorials feasible, the model is dependent on the availability of that package. One solution would be to use Stirling's approximation [135]

$$1 \cdot 2 \cdot 3 \ldots n = \sqrt{2\pi n} n^n e^{-n} e^{\omega(n)}$$

where $\omega(n)$ is bounded by

$$\frac{1}{12\left(n+\frac{1}{2}\right)} < \omega(n) < \frac{1}{12n}$$

so for large n

$$1 \cdot 2 \cdot 3 \ldots n = \sqrt{2\pi n} n^n e^{-n} \tag{64}$$

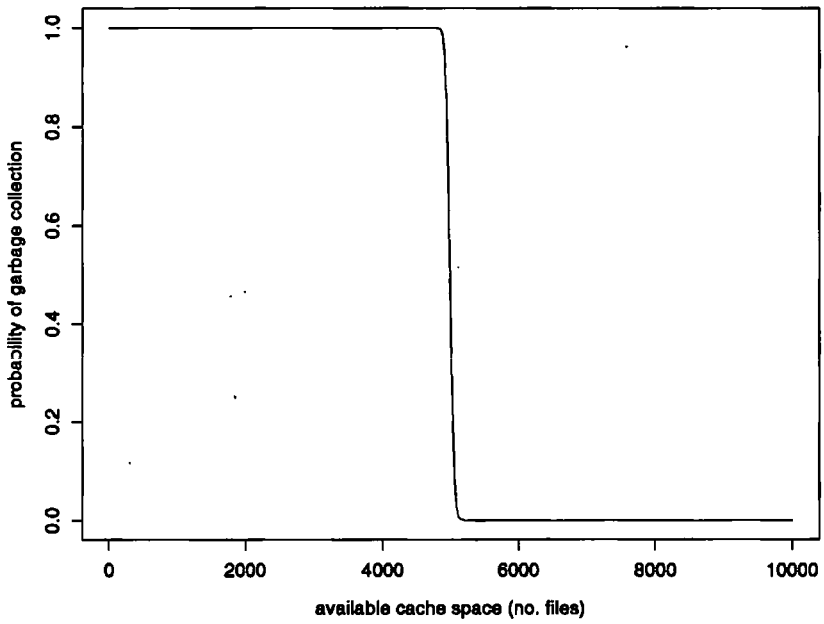
Using (64) and a tractable approximation for (63) can be found

(65)

$$P(Y \geq i) = n^{n+1/2} \sum_{k=i}^{k=n} \frac{p^k (1-p)^{(n-k)}}{\sqrt{2\pi k(n-k)} k^k (n-k)^{(n-k)}}$$

The graph in Figure 66 shows the probability distribution function (65), that is the probability that i files in n = 10,000 requests will result in a cache miss given that the cache miss rate is $p = 0.5$. So if i files are cleared out of the cache at the start of the day and during the day that the proxy receives 10,000 requests, then the graph in Figure 66 gives the probability that the garbage collector will run.

Figure 66 Garbage collection probability distribution ($p = 0.5$)



Time to compute (65) for $i = 0,1,...$ 10,000 was 404 seconds. However, it can be seen from the graph that (65) is a tightly centred around the mean np. That is, when n is large (65) the transition from $P(Y \geq i) = 1$ to $P(Y \geq i) = 0$ is steep and occurs over a relatively small range of i. Computing (65) over a smaller range of i would significantly reduce the computation time.

5.2.2 Data Analysis

The benefits of assuming that cache references are modelled by independent Bernoulli trials is that the mathematics required to develop an analytical model of increases in cache occupancy is fairly straight forward. As users of a World-Wide Web proxy server operate independently it would be reasonable to assume that cache references are uncorrelated. This assumption is investigated by analysing a sample week of cache references from the log files of an actual proxy server.

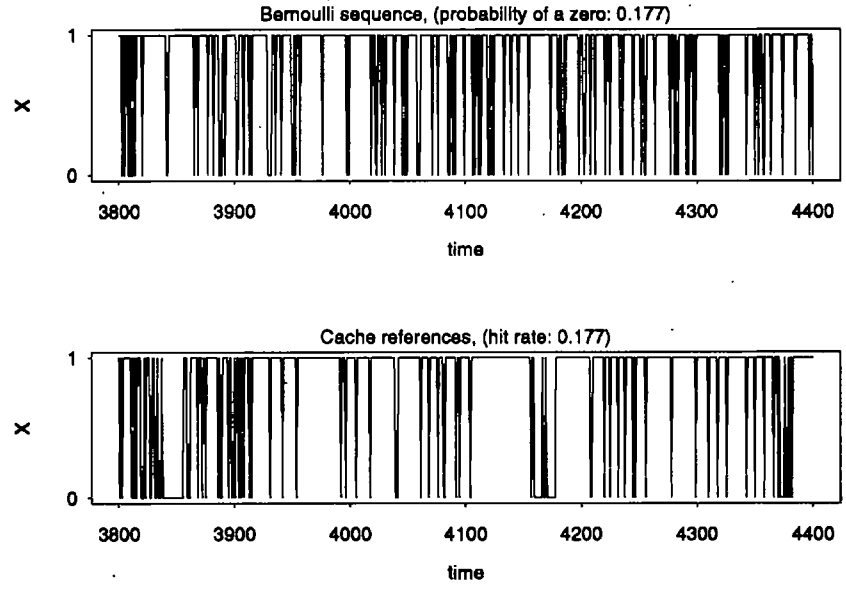
The sequence of cache references form a stochastic process $X = \{X(t)\}$, where $t = 0$, 1,..., and

$$X(t) = \begin{cases} 0 & \text{if cache hit} \\ 1 & \text{if cache miss.} \end{cases} \quad (66)$$

The cache hit or miss probabilities are denoted by $P(0)$ and $P(1)$ respectively, where $P(0)+P(1) = 1$. The cache hit and miss probabilities estimated from the log file sample were found to be $P(0) = 0.177$ and $P(1) = 0.823$.

Figure 67 shows an extract from a sequence of independent Bernoulli trials (for $P(0) = 0.177$) and an extract from cache references from the example week.

Figure 67 Time series extract of Bernoulli trials and cache references



The autocorrelations shown in Figure 68 of these cache hit/miss sequences appear to decay hyperbolically and are therefore long-range dependent unlike the Bernoulli trials. According to [223], autocorrelations of time series that decay roughly hyperbolically exhibit long-range dependence. The autocorrelation ρ_k is defined in terms of the autocovariance γ_k

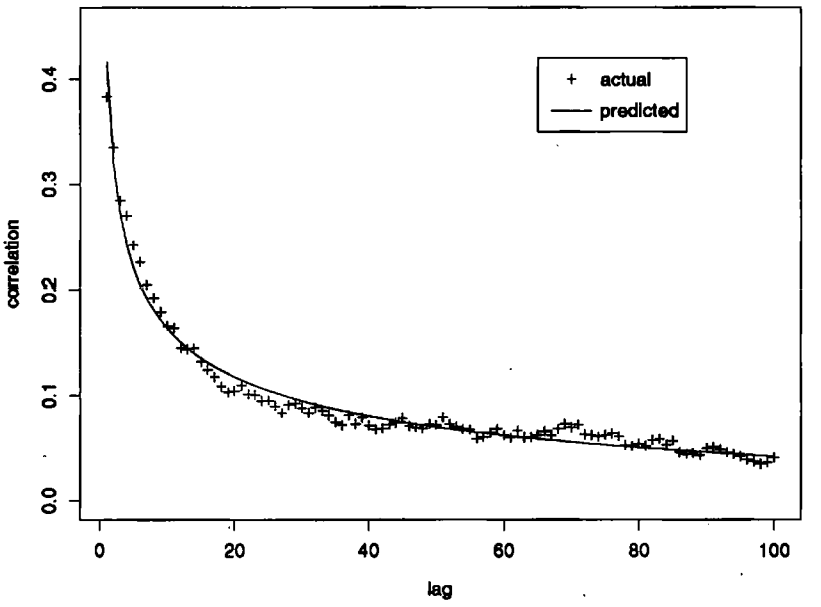
$$\rho_k = \frac{\gamma_k}{\gamma_0} \tag{67}$$

So for long-range dependent series, ρ_k can be modelled as

$$\rho_k = c_\rho k^{-\alpha_\rho} + b_\rho \tag{68}$$

where c_ρ is some constant and $0 < \alpha_\rho < 1$. Using non-linear regression modelling techniques [45] values of the parameters in (68) were estimated to be $c_\rho = 0.49$, $\alpha_\rho = 0.32$ and $b_\rho = -0.07$ derived from actual cache data.

Figure 68 Autocorrelation of cache references



A parameter often used to characterise long-range dependence is the Hurst parameter. Averaging series X over non-overlapping blocks of size m, results in a series of ensemble averages $X(t)^{(m)}$, which form a new stochastic process for each $m = 1,2...$, $^{(m)} = \{X(t)^{(m)}\}$. A process is said to be short-range dependent if, for

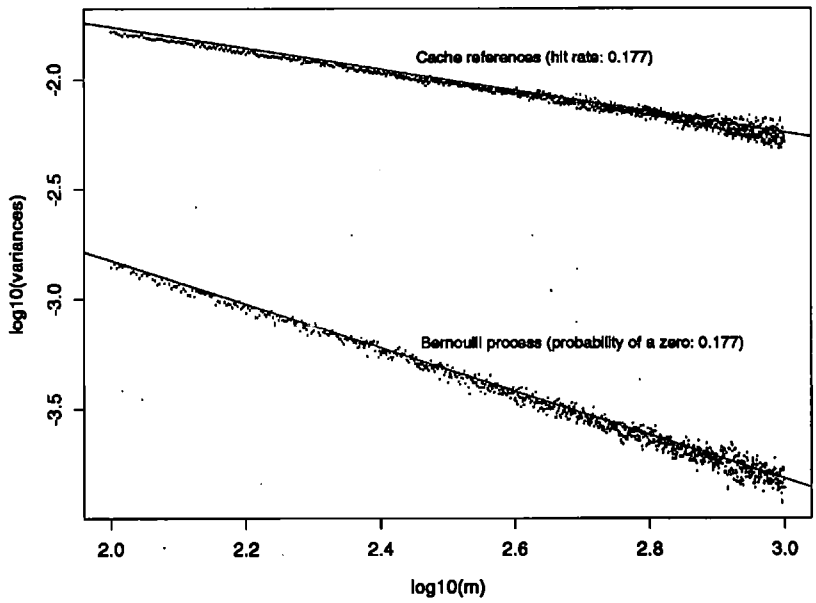
$$\mathrm{var}[X^{(m)}] \approx m^{-\beta} \qquad m \to \infty \tag{69}$$

$\beta = 1$. If, however $0 < \beta < 1$, the process is said to be long-range dependent. Taking logarithms of both sides and rearranging gives

$$\log_{10}(\text{var}[X^{(m)}]) \approx -\beta \log_{10}(m). \tag{70}$$

Figure 69 shows the variance-time log-log plot of the cache hit/miss sequences. It also shows the variance-time plot of a sequence of randomly generated Bernoulli trials of 0s and 1s for $P(0) = 0.177$.

Figure 69 Variance time plot



Their linear regression curves are shown as solid lines and the slope of these lines is related to the parameter β. The relationship between β and the Hurst parameter H is given by

$$\beta = 2 - 2H. \tag{71}$$

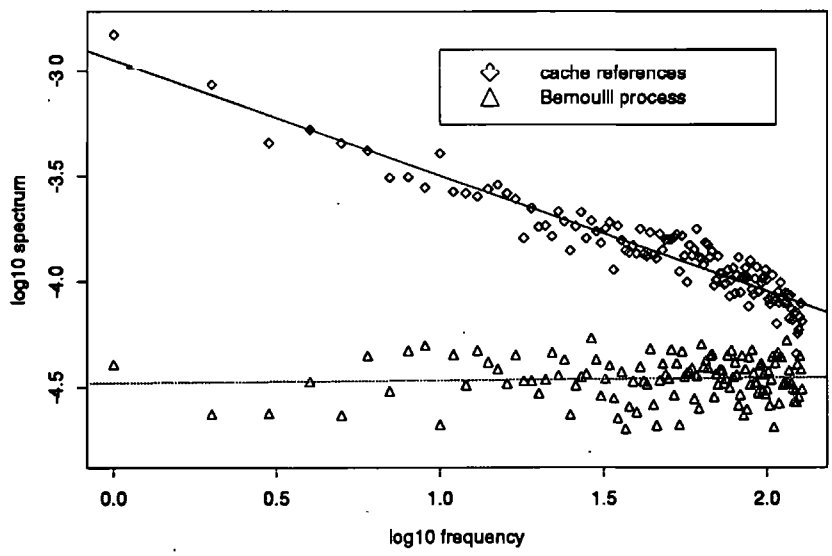
If $H = 0.5$ (that is when $\beta = 1$) then X is short-range dependent. However, for $H \in (0.5, 1)$ (that is when $0 < \beta < 1$), X is self-similar. For the randomly generated

Bernoulli trials, the slope of the linear regression curve was -0.996 resulting in a self-similar parameter of $H = 0.502$. The slope of the cache hit/miss sequence linear regression curve was -0.48, and hence $H = 0.76$ indicating significant long-range dependence. Self-similarity can also be determined in the frequency domain. The power spectrum $s_X(f) = \sum_k \rho_X(k) e^{2\pi i k f}$ resembles:

$$s_X(f) = |f|^{-\gamma} \tag{72}$$

for $0 < \gamma < 1$. Figure 70 shows the power spectrum for proxy server and a sequence of Bernoulli trials. The slope of the proxy cache references curve was $\gamma = 0.55$. Given that $H = (1+\gamma)/2$, the Hurst parameter was computed from the estimate of γ to be $H = 0.78$. For the Bernoulli trials $\gamma = 0.01$ resulting in $H \approx 0.5$.

Figure 70 Power spectrum



5.2.3 Fractional ARIMA Processes

In order to produce a realistic distribution of Y, $X(t)$ must be a long-range dependent process. Fractional autoregressive-integrated moving averages (ARIMA) [89][146] are

a family of processes that are long-range dependent. Given that ε_t is an independent gaussian random variable, a fractional ARIMA(p,d,q) process is defined by the equation

$$\phi(B)\nabla^d x_t = \theta(B)\varepsilon_t \tag{73}$$

where

$$\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p \tag{74}$$

and

$$\theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q \tag{75}$$

given that B is a back shift operator, and p and q are the respective orders of the autoregressive (AR) and moving average (MA) processes. ∇^d is the fractional differencing operator [111], such that

$$\nabla^d = (1 - B)^d \tag{76}$$

The parameter d is real and lies within the unit interval [0,1], therefore ∇^d can be derived from the following binomial expansion

$$\nabla^d = (1 - B)^d = 1 - dB - \frac{1}{2}d(1-d)B^2 - \frac{1}{6}d(1-d)(2-d)B^3 - \ldots \tag{77}$$

Long-range dependent cache references will be modelled using an ARIMA(0,d,0) process, that is $p = 0$, $q = 0$, therefore

$$\nabla^d x(t) = \varepsilon_t \tag{78}$$

The computation of (78) requires the generation of the sequence of Gaussian random variates ε_t. There are a number of ways in which random Gaussian variates can be generated from uniform (pseudo) random number generators (see [123], [197] and [198]). The results of the Monte Carlo simulations are dependent upon the quality of the generator. A flawed generator could give misleading results. It is therefore necessary to

ensure that the sequence of random numbers (both Gaussian and uniform) are sufficiently random and conform to their respective distributions. In Appendix B the statistical tests performed on these generators are described. Appendix B also discusses issues of period, performance and portabiltiy.

5.2.4 Monte Carlo Simulation

Monte Carlo simulation techniques use probabilistic methods in order to evaluate non-probabilistic expressions [123]. Using the Law of Large Numbers the probability of an event occurring can be estimated from its relative frequency of occurrence in repeated trials of a Monte Carlo simulation [121]. In this section Monte Carlo simulation is used to calculate the probability that the cache size Y grows up to and beyond a threshold i, $P(Y \geq i)$.

Now $x(t)$ is a continuous random variable with a mean centered around zero. $X(t)$ is a discrete random variable (0 or 1) centered around a mean of $p = P(1)$. So $X(t)$ is then given by

$$X(t) = \begin{cases} 0 & \frac{x(t)}{\max[|x(t)|]} + p < 0 \\ 1 & \frac{x(t)}{\max[|x(t)|]} + p \geq 0 \end{cases} \quad (79)$$

A value of $P(1) = 0.75$ was estimated from the log files over a 19 month period (March 1995 to October 1996). The fractional differencing parameter d is related to the Hurst parameter H and is given by

$$d = H - 1/2 \quad (80)$$

H was found to be 0.83 for the samples chosen and this results in $d = 0.33$ for this period. This data was applied to a model that assumed that a proxy server receives $n = 10{,}000$ requests per day which result in either a cache hit or a miss.

For values of $p = 0.75$ and $d = 0.33$ the fractional ARIMA process was used to generate 1,000 replications of $n = 10{,}000$ cache hit/miss sequences, that is the random variable $X_r(t)$ where $r = 1,\ldots,1000$. Given that a cache hit is represented by $X_r(t) = 0$ and a cache hit is represented by $X_r(t) = 1$ the random variable y_r represents (daily) growth in cache occupancy (in number of files) of the r^{th} replication of the simulation, where y_t is given by

$$y_r = \sum_{t=0}^{t=n} X_r(t) \tag{81}$$

An estimate of $P(Y \geq i)$ can therefore be computed from

$$P(Y \geq i) = \frac{1}{\max(r)} \sum_{r=1}^{r=\max(r)} y_r \tag{82}$$

The probability distribution curve $P(Y \geq i)$, is shown in Figure 71 (dotted line). The solid line in Figure 71 shows the probability curve derived for the analytical model (65) for $p = 0.75$, where (65) was based on the assumption that $X(t)$ can be modelled on independent Bernoulli trials.

While the distribution of increases in cache occupancy for Bernoulli trials is tightly centered around the expected mean of np, there is considerably more variation in the distribution derived from the ARIMA simulation.

It is interesting to note that if $X(t)$ is generated from uniformly distributed random number generator (that is, independent Bernoulli trials) the resulting frequency distribution of Y_r yields a probability distribution curve $P(Y \geq i)$ very similar to that given by (65).

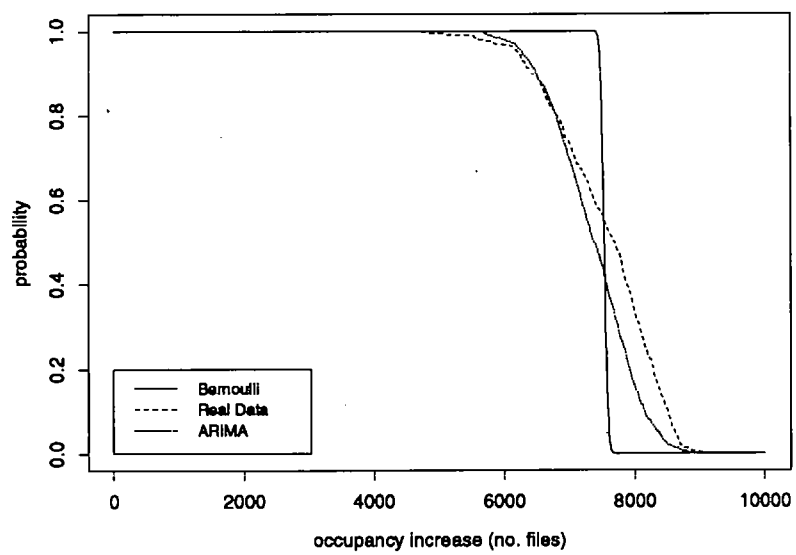
A third experiment was carried out where $X(t)$ was generated from the proxy server log files. Over the 19 month period more than 6 million cache references were made. These references were summed over sequential blocks of 10,000 to produce a sequence of Y_r.

The probability distribution curve $P(Y \geq i)$ derived from actual cache references is shown in Figure 71 (dashed line).

Cache growth as predicted by the Bernoulli model is different from the growth of an actual cache. This analysis illustrates that cache growth cannot be predicted merely from the cache hit rate and the assumption that cache references are independent.

It can be seen that the simulation model using the fractional ARIMA process produces a closer approximation of the distribution curve of the real data than that of the initial model using independent Bernoulli trials. This implies that cache references generated by a Bernoulli is less representative than using one that is long-range dependent and self-similar like a fractional ARIMA(p,d,q) process.

Figure 71 Probability distribution curves for increases in cache occupancy



It should be noted that alternative methods of generating long-range dependent, self-similar time series exist. However a ARIMA(0,d,0) process was used because it pro-

vides a flexible means of modelling varying degrees of self-similarity from a single parameter *d*.

5.3 Cache Trace-Driven Simulation

Trace-driven simulations have been used extensively in studying caches (for example Arlitt et. al. [6], Baentsch et. al. [10] and Markatos [157]). Trace-driven simulations were used here to investigate the effects of long-range dependence properties on cache performance and their influence on the choice of caching policies. The type of cache that was simulated was a main memory cache like one proposed by Markatos [157]. The simulation of the memory cache was implemented using the high level programming language C [132].

The analysis in section 5.2.2 showed that cache references do not occur independently, but exhibit long-range dependence and self-similarity. This section sets out to show that the effect of caching policy on the performance of the cache varies according to the presence or absence (independence) of long-range dependence.

The effects of long-range dependent Web references were investigated by using two trace data sets. The first data set was taken from a weekly log file of a Lucent proxy server (for the period 27 March through to 2nd April 1995). Each reference was submitted to the cache simulation in the (chronological) order it was received by the proxy. The second data set was taken from the same weekly log file but instead each log entry was randomly sorted so as to remove any long range dependence properties.

Main memory caches are typically small so replacement policies are an important design choice. Two replacement policies were examined:

- Least Recent Used (LRU)
- Least Frequently Used (LFU)

Garbage collection was carried out when a requested document would not fit into the available memory of the cache. Files were removed (dictated by the policy being used) until there was enough space to store the requested document). If the size of the requested document is larger than the actual cache size then the document is not cached.

5.3.1 Cache Performance

Web caches are used to improve network performance. However the system on which the cache operates becomes performance factor. A system with inappropriate CPU, disk or memory capacity can present a greater performance bottleneck than the network itself.

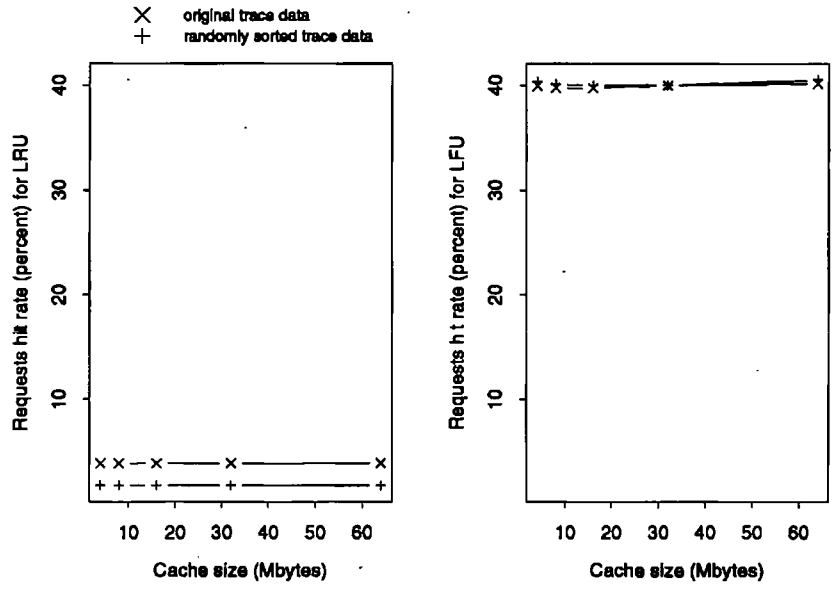
For this reason two aspects of performance were considered. The first was cache hit rate, in terms of both requests and bytes. High hit rates mean less network traffic volume. The second was garbage collection frequency. Garbage collection can consume system resources and therefore it is beneficial to avoid it where possible.

This section describes the results of the trace simulations of the memory cache for the original (unsorted) cache references and the randomly sorted cache references.

5.3.1.1 Cache Hit Rates

Cache performance was measured for cache sizes of 8, 16, 32 and 64 Mbyte using both Request hit rate and Byte hit rate. The two graphs in Figure 72 show the Request hit rate for both replacement policies.

Figure 72 Request percentage hit rate

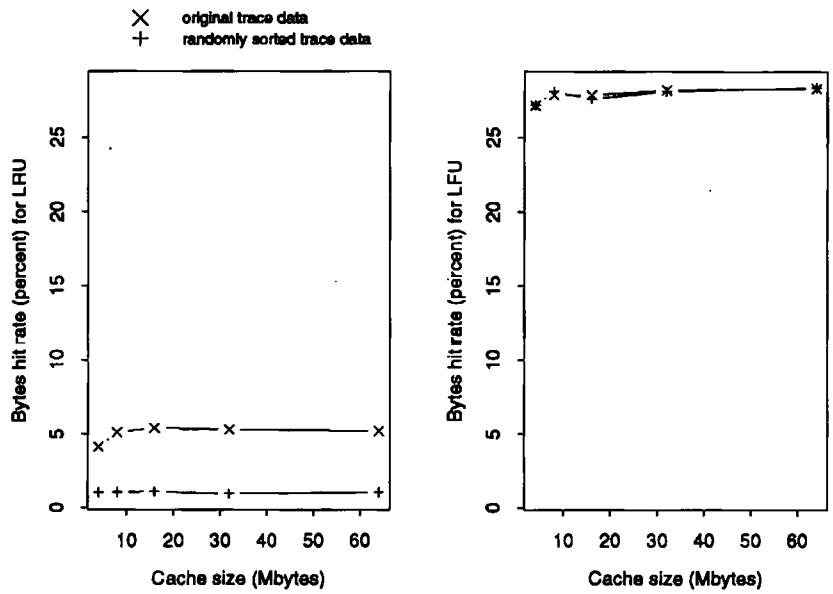


It can be seen that using a Least Frequently Used policy yields a much higher cache hit rate than a Least Recently Used. It is interesting to note that the in both cases the cache performance does not increase with cache size.

The most significant result is the effect of long-range dependence. For LFU replacement policy there is very little difference between the results of the original data set (long range dependent) and the randomly sorted data (independent). LRU does exhibit some difference between the two data sets, but the difference is still small.

The two graphs in Figure 73 show the Byte hit rate results.

Figure 73 Bytes percentage hit rate

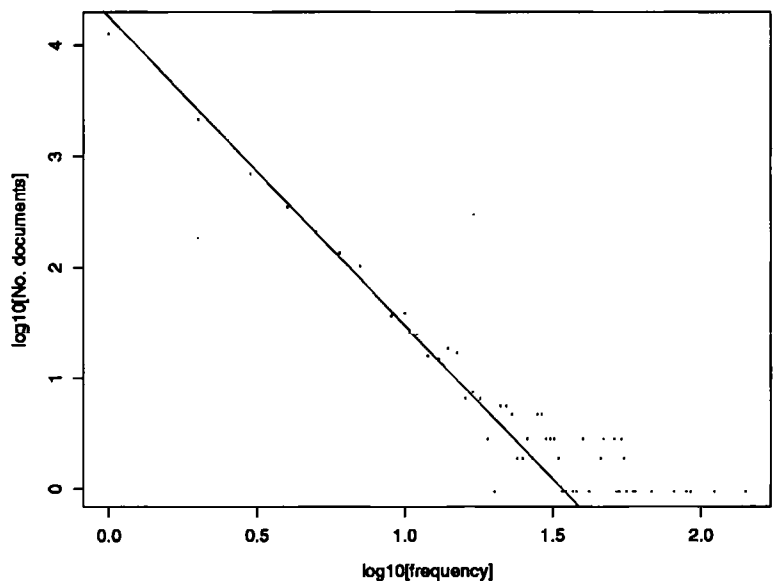


The Byte hit rate results are similar to that of Request hit rate. LFU out-performs LRU. Arlitt and Williamson [6] states that the superior performance of LFU over LRU is due to the general lack of strong temporal locality in Web documents references. The reason for this being that "client-side caching mechanisms remove temporal locality from the reference stream seen at the server" [6].

Figure 74 shows the Web document access statistics for the week ending 2nd April 1995. The number of documents that were accessed only once (that week) was 13,543. This meant that the number of "one timers" constituted 76% of the total requested documents (and 63% of the total bytes requested).

Given that there are a high number of "one time" references resulting in a cache "cluttered with useless files", there is a need for "techniques to expunge such files from a cache" [6].

Figure 74 Web document reference statistics



5.3.1.2 Garbage Collection

Figure 75 shows how frequently garbage collection takes place for LRU replacement policy. For small cache sizes garbage is collected more frequently when the references are long-range dependent (original trace data) compared to when references are independent (randomly sorted trace data). However when cache sizes are larger long-range dependence is not so significant.

Figure 75 Garbage collection frequency for LRU replacement policy

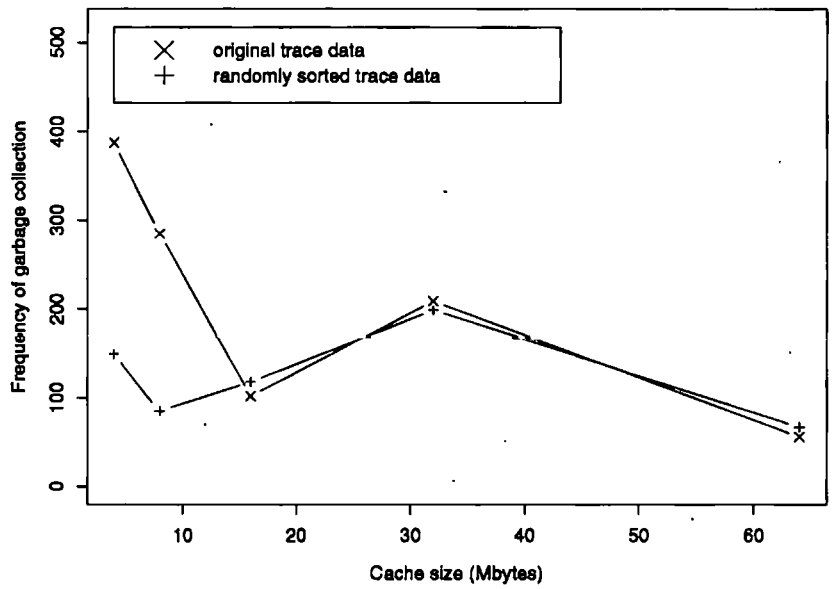
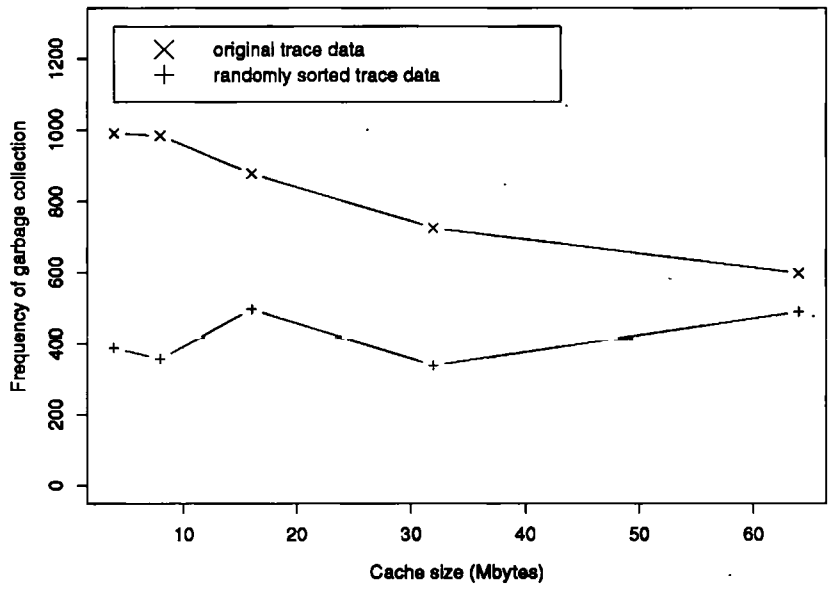


Figure 76 shows how frequently garbage collection takes place for LFU replacement policy. It can be seen that garbage collection is much higher for LRU compared to LFU.

However the most interesting result is that in garbage collection frequencies are much higher when it is assumed that references to the proxy are long-range dependent compared to independent references.

Furthermore independent references yield a fairly constant garbage collection frequency over changes in cache size, whereas when references are long-range dependent garbage collection frequencies fall as the cache size increases.

Figure 76 Garbage collection frequency for LFU replacement policy



5.3.1.3 Results Summary

The results from the above experiment have given some useful insights into garbage replacement policies and cache sizes. LFU favours a higher cache hit rate while LRU favours a lower garbage collection frequency. Both are desirable.

Now LRU cache hit rates are so poor that one wonders whether the cache is having any benefit or is even detrimental to network performance. That being the case LFU seems the preferred option but it must be accepted that higher garbage collection frequencies will result. Long-range dependence seems to be a factor here but cache references cannot be randomly sorted prior to submission to the proxy.

Garbage collection will consume system resources so investment here will help. The graph in Figure 76 shows that garbage collection frequencies diminish with cache size.

But there maybe other policies that can be adopted in order reduce garbage collection and enhance (or at least maintain) cache hit rates.

5.3.2 Collect Ahead

As described above when Garbage Collection takes place only a minimum number of files are removed in order to cache the next document. This means that after the cache's initial "cold start" it will be constantly near its capacity and that there will be a high probability that the cache will need to garbage collect whenever a cache miss occurs. The author calls this policy "Collect on Demand". An alternative to Collect on Demand is "Collect Ahead". This is where when the cache garbage collects, instead of removing just enough files to accommodate the current cache miss, it clears out space sufficient to accommodate future cache misses.

In this section Collect on Demand is compared to two Collect Ahead policies:

- remove documents until 25% of the cache is unoccupied
- remove documents until 1 Mbyte of the cache is unoccupied

The graphs in Figure 77 and Figure 78 show the Request and Byte hit rate results respectively.

Figure 77 Percent request hit rate with *Collect Ahead* (LFU)

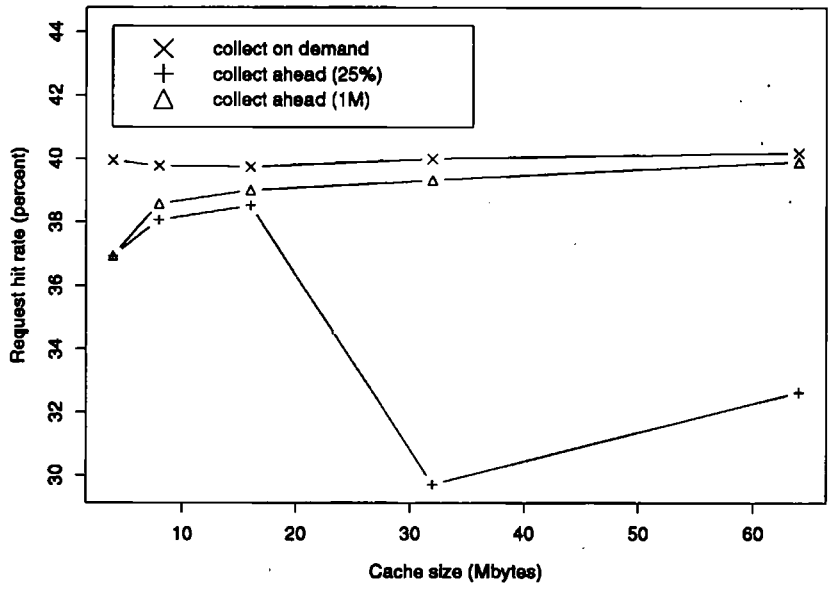
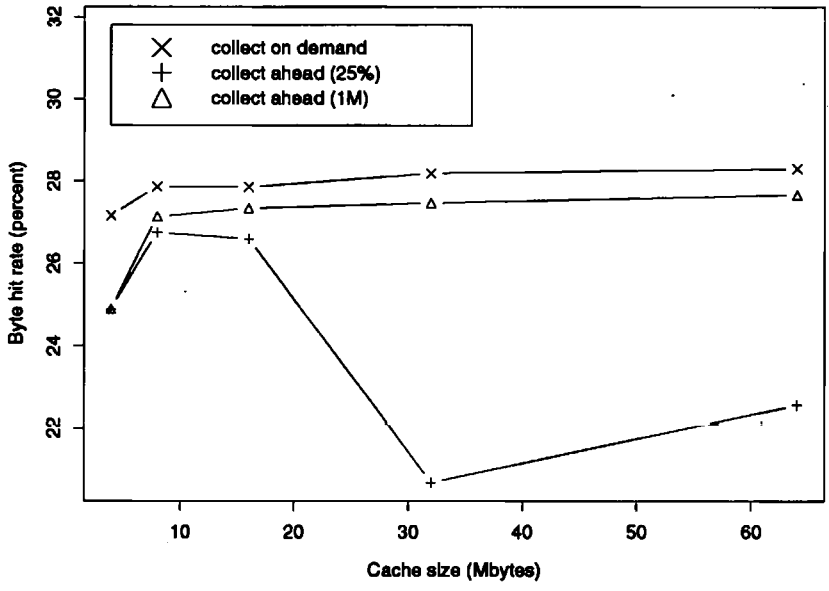


Figure 78 Percent byte hit rate with *Collect Ahead* (LFU)

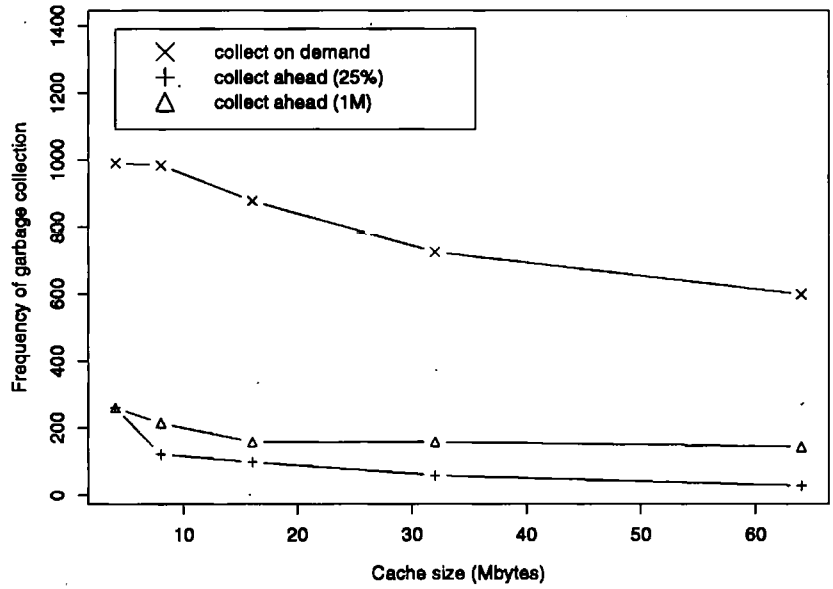


One would expect the probability of a cache hit to diminish with amount of cache occupancy. With Collect Ahead 1Mbyte the reduction in cache hit rate (Request and Byte) is

relative small. With Collect Ahead 25% however the reduction in cache hit rate is quite high for cache sizes greater than 16Mbytes. This illustrates the effects of removing too many files from the cache. From previous analysis the average Web document size is relatively small (median document size approximately 2Kbytes). This means when the cache size is large, many files have to be removed in order to clear such a significant percentage of space. These results suggest that this is too much.

The graph in Figure 79 shows the Collect Ahead results for Garbage Collection Frequency.

Figure 79 Garbage collection frequency with *Collect Ahead* (LFU)



It can be seen that Collect Ahead policies significantly reduces Garbage Collection frequencies. While Collect Ahead 25% brings about the best results in they are not significantly better than Collect Ahead 1Mbyte. Given the effects on hit rate when using Collect Ahead 25%, clearing out a small fixed amount would seem the best policy to adopt.

5.3.3 Prefetching

The concept of prefetching and its associated advantages/disadvantages were discussed in section 5.1.3. In this section the effects of prefetching on cache hit rate and garbage collection are examined using trace-driven simulations.

A very simple prefetch policy is adopted. This particular analysis concentrates on a main memory cache (rather than a disk cache). This means that the cache is cleared out when the proxy server is rebooted. It is assumed that the proxy server is rebooted periodically for maintenance purposes, say once a week. Therefore once a week the cache goes through its "cold start" process. This cold start process could be accelerated by prefetching documents during an outside prime time period. The following prefetch policies issues need to be addressed:

- what documents to prefetch
- how many documents to prefetch

The best documents to prefetch would be those that would be most popular for the coming week. But such a thing is difficult to predict. So some assumption of temporal locality of reference is made and the most frequently accessed documents from the previous week are prefetched. This policy of "warming up the cache" proposed by Baentsch [10]. The trace-driven simulation is repeated for varying amounts of prefetching.

The cache hit and garbage collection frequency results are shown in Figure 80 and Figure 81 respectively.

Figure 80 Cache hit rates with prefetching

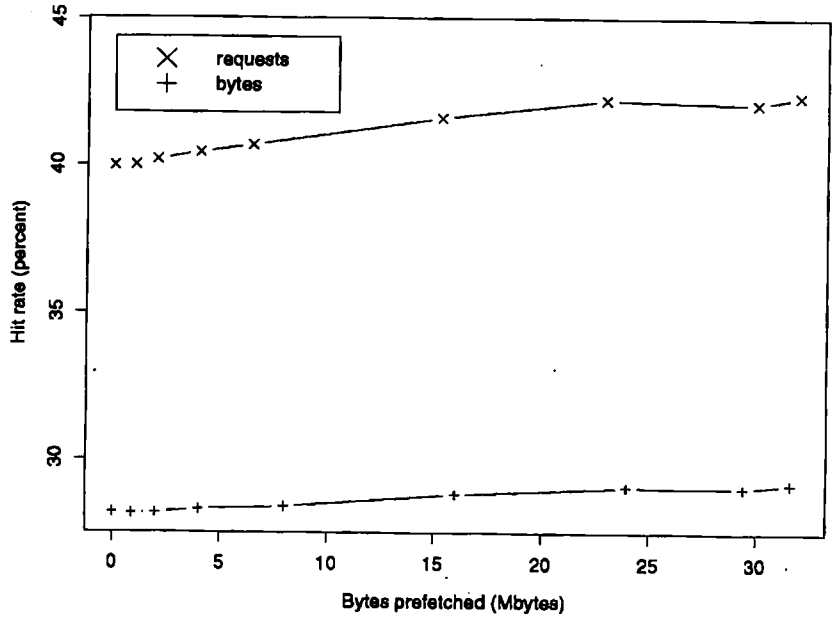
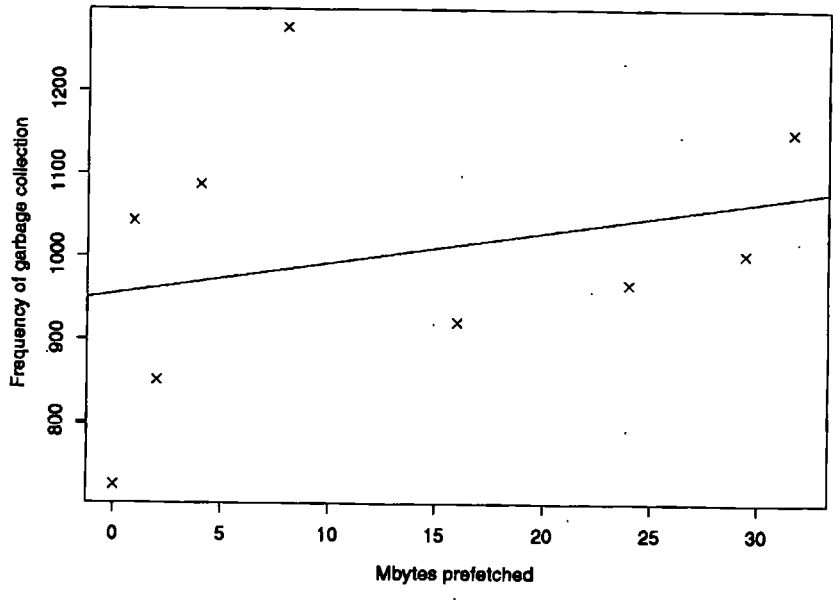


Figure 81 Garbage collection frequency with prefetching



The first point of the hit rate curves represents the results zero bytes prefetched (that is caching *without* prefetching). It can be seen that there is an improvement in cache hit

rates with the amount of prefetching. However the improvement is rather small. The effects on garbage collection are quite varied but there appears to be a positive correlation the amount the prefetching.

It would appear that "warming up the cache" each week does not bring about significant improvements to cache hit rates but can bring about increases in garbage collection frequencies. It is likely that the effectiveness of prefetching is dependent upon the ability to successfully predict future document access.

5.4 Chapter Summary

The technique of mirroring is based on a very simple user access model, that is some Web (or FTP) sites are more popular with users than others. While this is true a more refined model exists based upon the temporal locality of reference of individual document accesses. This model lends itself to caching technology as a replication technique.

Documents accesses incur higher propagation delays when fetched from a remote server (cache miss) than from the cache (cache hit). High cache hits rates are therefore generally conducive to lower propagation delays and capping traffic volumes on long haul communications facilities. Caches however can only hold a fraction of the entire Web space so they must carry out garbage collection in order to accommodate future cache misses. The problem is that excessive garbage collection can deplete the cache occupancy such that cache hits rates are adversely affected.

This chapter set out to study the growth of cache occupancy in order to gain an insight into caching policies. By treating cache references as a sequence of Bernoulli hit/miss trials a cache growth model can be constructed using the Binomial distribution. However the validity of the model is reliant on cache references occurring independently. The analysis in 5.2.2 showed that the assumption of independence in cache references

cannot be assumed. However what is significant about the results in this chapter is that they show that the assumption of independence cannot ignored!

Cache reference were found to be highly correlated (long-range dependent/self-similar) such that periods of high miss rates would be common. This leads to periods of rapid cache growth resulting in frequent execution of the garbage collector if a conservative document purging policy is adopted. Indeed the trace simulations showed that when documents were purged "on demand" the presence of long-range dependence in cache references caused garbage collection to take place more frequently compared to when cache references were independent.

The development of an analytical model of cache occupancy increases was relatively straight forward provided the occurrence of cache references were assumed to be independent Bernoulli trials. However the results of the Bernoulli trial model were significantly different to cache growth measurements taken from data of an actual proxy server.

Monte Carlo simulation techniques proved to be a useful method of modelling cache occupancy growth. Long-range dependent, self-similar cache hit/miss sequences were generated using a fractional ARIMA(0,d,0) process, where d is the fractional differencing parameter which is related to Hurst parameter H. The Hurst parameter is a measure of the degree of self-similarity and it can be derived from cache reference data in either the time or frequency domain.

A probability distribution curve of cache growth was constructed by generating repeated sequences of cache hits and misses using a ARIMA(0,d,0) process for a value of d computed from real cache reference data. The cache growth predicted by this model was quite close to those measured from an actual proxy server.

Cache trace simulations showed that a Least Frequently Used replacement policy yields higher hit rates than a Least Recently Used policy. While the presence of long-range dependence in cache references (compared to independent cache references) does not appear to affect hit rate significantly, long-range dependence does result in higher garbage collection frequencies. Adopting a collect ahead policy helps to reduce garbage collection frequencies, but it should be conservative otherwise hit rates are significantly affected. Otherwise, if a collect on demand policy is used then the proxy server must have adequate CPU cycles and disk bandwidth in order to cope with the overhead imposed by the garbage collection process.

Analysis of the growth in cache occupancy can help determine which caching policies bring about the best cache performance. However accurate decisions cannot be made based upon the hit rate alone due to the presence of long-range dependence in cache references. In which case, not only will the magnitude of the cache hit ratio be a contributing factor to the performance of the cache, but also the degree of long-range dependence.

CHAPTER 6 *Conclusions*

This thesis looked at improving the performance of wide area networks. In CHAPTER 2 a model of packet delays was presented. Experimental results showed how framing delays can be reduced by increasing the speed of communication facilities. These results also show how significant propagation delays can be as the distance between communicating computer systems increases. Propagation delays are bounded by the speed of light and are not affected by increases in the speed of communication facilities. Therefore the performance benefits that are brought about by investments in bandwidth are subject to diminishing returns, especially as propagation delay and the cost of communication facilities increases with distance.

Wide area networks are predominantly packet switched because this is considered to offer efficient use of communication facilities (CHAPTER 1). However network resources are shared amongst users by multiplexing individual packet streams on to a common communication channel, which gives rise to contention. As the network gets busier the queues at communications channel interfaces grow which results in increased waiting times.

The expression in (6) showed that waiting delays are inversely proportional to the service rate μ, thus waiting delays can be reduced by upgrading communication facility speeds. However the expression in (6) is also related to the traffic utilisation ρ and traffic utilisation is a function of the demand placed on the network by the user community.

Users of the Internet do not incur costs related to access duration, traffic volumes, distance or time of day. There is therefore little incentive for a user to quench their demand for network resources. The reason why a network like the Internet can become congested is due to the (current) lack of congestion control methods. Free of any (further) financial constraints, resource management policies are almost entirely under the control of the end user.

Network performance is therefore an ethical issue as well as a technical one. Game Theory was used to model network users as independent interacting decision makers. Network usage was modelled most appropriately on the Volunteer's Dilemma, whereby a "common good" is brought about by the voluntary actions of one or more members of a community. In a network community each user adopts a network access "strategy". Users can "cooperate" towards the network common good by regulating their access so as not to congest the network. Any individual user adopting this strategy lowers their own personal productivity for the benefit of others. Alternatively some users may "defect" against the common good in order to maximise their own productivity. Minority defectors are called "free riders" (in Game Theory terminology) in that their high productivity gains are brought about by remaining majority adopting a cooperative strategy. Majority defection is often referred to as the "tragedy of the commons" whereby a common resource, in this case the network, is subject to "overgrazing".

In order to overcome this the payoff matrix needs to be altered such that the dilemma is eliminated. One way would be to increase bandwidth resources so that they exceed demand. However, data network traffic tends to very bursty and therefore engineering

the capacity of the network in order meet peak demands can result in communication facility speeds that are excessive compared to the average amount of traffic.

Alternatively access restrictions could be introduced in order to regulate traffic. One method could be to adopt codes of conduct whereby users are expected to access the network in way that is beneficial to the common good. However, this would be notoriously difficult to implement and highly political. Another method would be to impose economic constraints on the use of network resources. But this too is a contentious issue as it creates an "elitist" access policy based upon ability to pay.

The network community is growing, both in terms of size and its appetite for data access. If demands for resources go unchecked the investment in bandwidth necessary to meet these demands could be prohibitively high, particularly for wide area access. Directed by the delay model developed in CHAPTER 1 this thesis has presented research into Compression, Mirroring and Caching as alternative (or complementary) methods of improving wide area network performance. These are not congestion control methods and so cannot avert the tragedy of the commons. However, the performance of wide area networks can be improved without incurring the high costs associated with bandwidth upgrades.

6.1 Compression

Data compression is an established method of optimising wide area network performance. Data compression can be done either at the application or in the network itself, but not both. The decision as to which is best, or if either are beneficial, is not an easy one.

When transmitting data across a network the compression process is being carried out in "real time" and so contributes to the transmission delay. Any benefits brought about by compression are bounded by the processing capabilities of the device executing the compression process. The experiments carried out in CHAPTER 3 showed that for low

link speeds compression brought about improved productivity but for higher speeds the processing delay incurred was such that compression was not beneficial (or could even be detrimental). Clearly processing delays caused by the compression process can be reduced by investment in processing resources.

The benefit of doing compression in the network is that it is under the control of the network administrators and designers. While the facilities to do compression may exist at the application level, users and may not always utilise them.

In the network packet compression is done on a generic stream of data (packet or data interleaved from many sources). Furthermore users may send data that cannot be compressed such as certain kinds of encrypted data or data types where compression is inherent within the encoding scheme. All this contributes to reducing the susceptibility of the data to compression. By doing compression at the application level schemes and models can be adopted that are appropriate the data type, thereby yielding higher compression ratios.

Network designers and administrators are therefore faced with the trade-off between having control over the compression process and achieving high compression ratios.

6.2 Replication

Productivity gains can be brought about by adding a mirror site. Mirrors are good for wide area network performance because they reduce access latencies and cap traffic loads on "long haul" communication facilities. These productivity gains increase as the distance between the mirror location and the user location is reduced. Furthermore communication facility costs are directly proportional to distance so it is possible to have a faster link to the mirror than the remote server.

The delay model in CHAPTER 2 was developed further in CHAPTER 4 to examine the performance benefits of using remote servers with mirrors. The analysis showed the performance benefits of locating the mirror site closer to the user (than the remote). Despite the reduction in propagation delays the benefits of using a mirror located to the user are small for low bandwidth communication facilities. This is because the overall delay is dominated by waiting delays. For high bandwidth links the propagation delay becomes more significant when users access distant remote servers and will therefore benefit from accessing a closer mirror.

A surprising result from this analysis showed that, regardless of the difference between bandwidth and proximity, the optimum productivity always occurred when the load on the remote and mirror was approximately evenly balanced. If the mirror is far closer than the remote and has a much higher bandwidth communication facility, then the mirror users will have a far superior performance (productivity) than the remote users. Migrating remote users onto the mirror (while improving their performance) diminishes the performance of the existing mirror users. Furthermore the network no longer function at the optimum productivity, Thus, there is a dilemma between operating at the optimum social productivity and avoiding a two tier level of service.

Mirrors are not considered by the industry to be effective methods of improving wide area network performance. The Teamwork experiment showed how response times were reduced and traffic volumes capped through replication techniques. But more importantly it showed that an analysis of how the users accessed the remote data helped to bring about these performance improvements. Investment in resources for improving performance can be misguided if user access models are inappropriate.

The mirror concept is based upon a user access model of site popularity. Typically however, only a few documents are ever accessed frequently. In which case, given that the

majority of documents are accessed infrequently, replicating a remote server in its entirety is not going to be that beneficial.

Caching employs "temporal locality of reference" as its model for replicating documents and is deemed a more effective replication method compared to mirroring. However caching as a replication method is less straight forward than mirroring. Caches are limited in size so they must adopt policies which "select" documents to replicate. Garbage Collection and Replacement policies determine which documents should be purged when the cache reaches its occupancy limit. In terms of wide area network performance, the objective of these policies is to maintain as high a cache hit rate as possible. However it is necessary to avoid excessive garbage collection as this will result in a high processing delays.

In CHAPTER 5 a study of cache growth was carried out in order to determine the most effective caching policies. An analytical model was developed to predict cache growth based upon references occurring as independent Bernoulli trials. The assumption of independence makes the development of such a model relatively straight forward.

However analysis of the log files of an actual proxy server revealed cache hit/miss sequences to be highly correlated and long-range dependent. In addition they exhibited properties of self-similarity. These properties make the use of conventional probability methods inappropriate for modelling cache growth. As a result Monte Carlo simulation techniques were used instead.

An ARIMA(0,d,0) process was used to generate long-range dependent self-similar cache references. While alternative methods of generating self-similar time series exist, an ARIMA(0,d,0) process was used because it provides a flexible means of modelling different degrees of self-similarity from a single parameter d, where d, the fractional differencing parameter is related to the Hurst parameter H. The Hurst H parameter is a

measure of self-similarity. It was shown how H is derived from cache reference data in either the time or frequency domain.

Monte Carlo simulations produced similar probability distribution curves for increases in cache occupancy for both the fractional ARIMA(0,d,0) process and real data. When cache hit/miss sequences are modelled on independent Bernoulli trials, the deviation from the mean occupancy increase is small which is not the case when hit/miss sequences that are long-range dependent. Given that actual cache references were found to be long-range dependent, in making the Bernoulli assumption, cache occupancy increases would be frequently underestimated and hence cache performance is over estimated.

The trace simulations of a proxy main memory cache showed that garbage collection is much more frequent when references are long-range dependent compared to independent references. The initial simulations did garbage collection *on demand*, that is, only enough space was cleared out of the cache to accommodate the next cache miss. On demand garbage collection keeps the cache close to capacity. The advantage of this is that the occupancy is kept as high as possible in order to maximise hit rates. The disadvantage is that there will be periods of frequent garbage collection. As the cache is always near its capacity limit there is a high probability that garbage collection will be required each time a cache miss occurs. Long-range dependence in cache references compounds this problem because there will be long periods of consecutive cache misses (more so than is they were independent).

Adopting a *collect ahead* policy reduces garbage collection. Collect ahead purges the cache of more files than is required to accommodate the next cache miss, thereby reducing the probability of requiring garbage collection for cache misses in the near future. The consequence of collect ahead is that cache hit rates are reduced. However, the trace simulations showed only a relatively small amount space has to be purged from the

cache in order to bring about a significant reduction in garbage collection frequencies, while keeping hit rates high.

6.3 Future Work

This thesis has examined methods of improving wide area network performance. These methods can bring about performance improvements, but as was shown in the analysis of mirror sites bottlenecks can still occur when demand exceeds capacity. In the (not too distant) future the Internet will be expected to support multimedia applications (video and audio) that have real time requirements. Merely improving performance may not be enough, rather wide area networks will be required to provide performance guarantees.

Currently packets are routed through the Internet on a "best effort" basis, there are no performance guarantees because the network cannot control the rate at which user generate traffic. When the utilisation of resources is high, the magnitude and the variance of delays are high.

Game Theory has proved to be a useful method of modelling user resource consumption. Given that the rewards for any individual user of a network are related to the amount of information they access then it is reasonable to assume that their consumption of network resources is likely to be high if they wish to maximise their own gains.

The Game Theory analysis showed that the "cost" of using the network to any individual user, during busy periods is imposed by the access behaviour of others. That is, the cost incurred by the users during periods of congestion take the form of increased access times. Typically the financial costs remains the same for periods of high and low usage. There is therefore, little incentive for any individual user to regulate their usage when the network is busy. In this environment it is difficult to support any "quality of service".

There is a current trend in network research literature to adopt a usage-sensitive pricing policy whereby users are impelled to "volunteer" to restrict their network access for the good of others through financial constraints. That is the rewards of being a "greedy" user (particularly during busy periods) diminish due to a financial cost imposed by the network provider rather than a performance cost imposed by other users.

However, in adopting usage-sensitive pricing, users will expect performance guarantees. This will require the introduction of new technologies. In order to support applications that require guaranteed bit rates the network must have some control over the load that is placed upon it. Traffic shaping technology provides a means for an application to notify the network of the nature of the traffic it is going to generate. The network can then assess whether it can support the requested traffic flow. The network will also monitor an application's traffic flow to ensure it adheres to the application's request. The network may drop packets of a flow that exceeds its stated parameters.

Resource reservation protocols are currently being developed to enable applications to signal to the network their performance requirements. Users will be charged accordingly for the resources they reserve. Adopting usage sensitive pricing to a network like the Internet will require cultural changes as much as it will need new technology. As the financial burden of the Internet migrates from the government (of the United States) to the private sector the over-provisioning of communication facilities as a means of addressing network congestion will not be acceptable. The Internet cannot continue to sustain an "all you can eat" pricing policy and the development of a new pricing model represents one of the greatest challenges.

References

[1] Abshire G.M., *Ethics Apply to Computer Specialists Too.* Computers and Society, Volume 11, Number 2, pp22, Spring 1981.

[2] Abshire G.M., *The Ethical Insensitivity of Computer Specialists and What You Can Do About It.* Computers and Society, Volume 12, Number 1, pp10-11, Winter 1982.

[3] Adobe Systems Incorporated. *PostScript Language Tutorial and Cookbook.* Addison-Wesley Publishing Company, 1991. ISBN: 0-201-10179-3.

[4] Agarwal A., Horowitz M. and Hennessy J., *An Analytical Cache Model.* ACM Transactions on Computer Systems, Volume 7, Number 2, pp184-215, May 1989.

[5] Andreessen M., *NCSA Mosaic Technical Summary.* Revision 2.1, marca@ncsa.uiuc.edu, May 1993.

[6] Arlitt M.F. and Williamson C.L, *Internet Web Servers: Workload Characterization and Performance Implications.* IEEE/ACM Transactions on Networking, Volume 5, Number 5, pp631-645, October 1997.

[7] Attanasio C.R., Markstein P.W. and Philips R.J., *Penetrating an Operating System: a Study of VM/370 Integrity.* IBM System Journal, pp 102-116, No. 1 1976.

[8] Baalke R., *Comet Shoemaker-Levey Home Page (JPL).* http://newproducts.jpl.nasa.gov/sl9/sl9.html, January 1995.

[9] Bach M.J., *Design of the Unix Operating System.* Prentice-Hall International Editions, 1986. ISBN: 0-13-201757-1 025.

[10] Baentsch M., Lauer A., Baum L., Molter G., Rothkugel S. and Strum P., *Quantifying the Overall Impact of Caching to Replication.* {baentsch, lauer, lbaum, molter, sroth, strum}@informatik.uni-kl.de, February 1997.

[11] Baentsch M., Baum L., Molter G., Rothkugel S. and Strum P., *Enhancing the Webs's Infrastructure: From Caching to Replication.* IEEE Internet Computing, Volume 1, Number 2, pp18-27, March 1997.

[12] Baentsch M., Baum L., Molter G., Rothkugel S. and Strum P., *World Wide Web Caching: The Application-Level View of the Internet.* IEEE Communications Magazine, pp170-178, June 1997.

[13] Baird B.J., Baird Jr. L.L. and Ranauro R.P., *The Moral Cracker?* Computers and Security 6, pp 471-478, 1987.

[14] Barns W., *Guidelines for Efficient Subscriber Usage of the Defence Data Network.* The MITRE Corporation, MTR-90W00007, January 1990.

[15] Bass T., Freyre A., Gruber D. and Watt G., *E-Mail Bombs and Countemeasures: Cyber Attacks on Availability and Brand Integrity.* IEEE Network, Volume 12, Number 2, pp10-16, March/April 1998.

[16] Becker R.A., Chambers J.M. and Wilks A.R., *The NEW S Lamguage*. Wadsworth & Brooks/Cole Advanced Books and Software, ISBN: 0-534-09192-X.

[17] Bel N.J.C., AT&T *Echo Canceller Enhances Customer Satisfaction*. Trends in Telecommunications, Volume 9, Number. 1, August 1993.

[18] Bell M., *Shape of the Future*. Network World, pp23-24, August 1997.

[19] Bell T., Cleary J.G. and Witten I.H., *Text Compression*. Prentice Hall Inc., 1990. ISBN: 0-13-911911-4.

[20] Benda M., *Turing's Legacy for the Internet*. IEEE Internet Computing, Volume 1, Number 6, pp75-77, November/December 1997.

[21] Bendat J.S. and Piersol A.G., *Random Data*. John Wiley and Sons, 2nd Edition, 1986, ISBN: 0-471-04000-2.

[22] Beran J., Sherman R., Taqqu M.S. and Willinger W., *Long-Range Dependence in Variable-Bit-Rate Video Traffic*. IEEE/ACM Transactions on Communications. Volume 43, Number 2/3/4, pp1567-1579, February/March/April 1995.

[23] Beran J., *Statistics for Long-Memory Processes*. Chapman and Hall, New York, 1994. ISBN: 0-412-04901-5.

[24] Berners-Lee T.J., Cailliau R., Groff J.F. and Pollermann B., *World-Wide Web: An Information Infrastructure for High-Energy Physics*. C.E.R.N, file:// info.cern.ch/pub/www/doc/www-for-hep.ps.Z, December 1994.

[25] Berners-Lee T.J., Cailliau R., Groff J.F. and Pollermann B., *World-Wide Web: The Information Universe*. C.E.R.N., file://info.cern.ch/pub/www/doc/ ENRAP_9202.ps.Z, February 1992.

[26] Bertsekas D. and Gallager R., *Data Networks*. Prentice-Hall International Editions. 1987, ISBN: 0-13-207674-5.

[27] Bestavros A., *Demand-based Documentation Dissemination to Reduce Traffic and Balance Load in Distributed Information Systems*. Proc. 1995 7th IEEE Symp. on Parallel and Distributed Processing, pp338-345, 1995.

[28] Bestavros A., *Using Speculation to Reduce Server Load and Service Time on the WWW*. Proceedings of the Fourth ACM International Conference on Information and Knowledge Management, pp28-32, November 1995.

[29] Birman K.P. and van Renesse R., *Software for Reliable Networks*. http://www.sciam.com/0596issue/0596birman.html, May 1996.

[30] Black U.D., *Data Networks Concepts, Theory, and Practice*. Prentice Hall, 1989. ISBN: 0-13-198599-X.

[31] Bloombecker J., *Computer Security - For the People*. Computers in Society, Volume 14, Number 4, pp12-15, 1985.

[32] Boggs D.R., and Mogul J.C., Kent C.A., *Measured Capacity of an Ethernet: Myths and Reality*. Computer Communication Review, Volume 18, Number 4, pp 222-234, Proceedings of the ACM SIGCOMM '88 Workshop, August 1988.

[33] Bosco H.L. and Gitlin R.D., *Overview*. Bell Labs Technical Journal: Packet Networking, Volume 2, Number 2, pp 3-4, Spring 1997.

[34] Boutell T., *How Can I Make Transparent and Interlaced GIFs and What are They?* http://boutell.nais.com/faq/tinter.htm, (last modified) 27 September 1995.

[35] Box, G.E.P., Hunter W.G. and Hunter J.S., *Statistics for Experimenter, An Introduction to Design, Data Analysis and Model Building*. John Wiley & Sons, 1975. ISBN: 0-471-09315-7.

[36] Brandscomb A.W., *Common Law for the Electronic Frontier.* Scientific American, Voume 265, Number 3, pp116-120, September 1991.

[37] Braun H. and Claffy K., *Pricing the Internet.* Public Access to the Internet, pp269-314, The MIT Press., 1995. ISBN: 0-262-11207-8.

[38] Brewer T., *Unix Benchmark System.* Naval Ocean Systems Center, Technical Document 1111, San Deiego California 92152-5000, July 1987.

[39] Brown W.J., *GCSE and "O" Level Law.* Sweet & Maxwell Ltd. London, third edition, 1986.

[40] Burger R., *Computer Viruses a High-Tech Disease..* Abacus, 1988. ISBN: 1-55755-043-3.

[41] Campbell A., *Remade on the Net.* .net, Issue 6, pp38-41, May 1995.

[42] Carlson B., Burgess A. and Miller C., *Timeline of Computing History.* Computer, Volume 29, Number 10, pp92, October 1996

[43] Cerf V.G., *Networks.* Scientific American, Volume 265, Number 3, pp42-51, September 1991.

[44] Chiao R.Y., Kwiat P.G. and Steinberg A.M., *Faster than Light?* Scientific American, Volume 269, Number 2, pp38-46, August 1993.

[45] Chambers J.M. and Hastie T.J. (editors), *Statistical Models in S.* Chapman & Hall Computer Science Series, 1996. ISBN: 0-412-05301-2.

[46] Cisco, *Cisco IOS Data Compression.* http://www.cisco.com/warp/public/538/1.html, 8 September 1997.

[47] Cisco, *Router Products Command Reference.* Internetwork Operating System Release 10, Chpaters 10 to17, Customer order number DOC-RTCF1, Text Part Number 78-1306-01, 1994.

[48] Cisco, *Internet Quality of Service.* Cisco White Paper, http://www.cisco.com/, 1997.

[49] Cisco, *How to Cost-Effectively Scale Web Servers.* Cisco, http:// www.cisco.com/warp/public/784/5.html, April 1998.

[50] Cisco, *Compression Service Adaptor.* Cisco, http://www.cisco.com/warp/public/733/adap/csa/index.shtml, 5 August 1998.

[51] Civille R., *The Internet and the Poor.* Public Access to the Internet, pp176-207, The MIT Press, 1995. ISBN: 0-262-11207-8.

[52] Claiborne J.D., *Mathematical Preliminaries For Computer Networks.* John Wiley & Sons Inc., 1990. ISBN: 0-471-51062-9.

[53] Clarke R., *Asimov's Law of Robotics: Implications for Information Technology Part 1.* Computer, pp51-61, December 1993.

[54] Clarke R., *Asimov's Law of Robotics: Implications for Information Technology Part 2.* Computer, pp57-66, January 1994.

[55] Cochran R., *The Power Within.* IEE Review, Volume 43, Number 5, pp220-222.

[56] Cochran W.T., Cooley J.W., Favin D.L., Helms H.D., Kaenel R.A., Lang W.W., Malling G.C., Nelson D.E., Rader C.M. and Welch P.D., *What is the Fast Fourier Transform*? Proceedings of the IEEE, Volume 55, Number 10, pp 1664-1674, October 1967.

[57] Cochrane P., *From Copper to Glass: the Right Idea, Decisions and Investments at the Right Time...* The Computer Journal, Volume 40, Number 1, pp1-11, 1997.

[58] Cohen S. and Grace D., *Engineers and Social Responsibility: An Obligation to Do Good.* IEEE Technology and Society Magazine, Fall 1994.

[59] Comer D.E., *Internetworking with TCP/IP.* 2nd ed. Volume 1, Englewood Cliffs. NJ: Prentice Hall 1991. ISBN 0-13-474321-0.

[60] Couch L.W., *Digital and Analog Communication Systems.* Macmillan Publishing Company, Third Edition, 1990. ISBN:0-02-325391-6.

[61] Crane E., *Finding Some Comfort on the Web.* Open Systems Computing, Volume 12, Number 2, pp90-91, February 1995.

[62] Corwella M.E. and Bestavros A., *Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes.* IEEE/ACM Transactions on Networking, Volume 5, Number 6, pp835-846, December 1997.

[63] Danskin J.M., *Compressing The X Graphics Protocol.* Princeton University Ph.D Thesis, jmd@cs.dartmouth.edu, January 1995.

[64] Danzig P.B., Hall R.S. and Schwartz M.F., *A Case for Caching File Objects Inside Internetworks.* ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/FTP.Caching-PS/Paper.ps.Z, University of Colorado, March 1993.

[65] Davies P. *About Time.* Viking, 1995. ISBN: 0-670-84761-5.

[66] Degan J.J., Luderer G.W.R. amd Vaidya A.K., *Fast Packet Technology for Future Switches.* AT&T Technical Journal, pp36-48, March/April 1989.

[67] Denton N., *Back to Broadcasting.* Financial Times, pp14, 15 Octoner 1997.

[68] Dertouzos M.L., *Communications, Computers and Networks.* Scientific American, Volume 265, Number 3, pp 31-37, September 1991.

[69] Dettmer R., *Almost an Accident.* IEE Review, Volume 44, Number 4, pp169-172, July 1998.

[70] Dick P.K., *Do Androids Dream of Electric Sheep?* Grafton, 1972.

[71] Drupsteen J., *ATM: the Leap to Broadband Services*. Trends in Telecommunications, Volume 2, Number 2, pp4-8, 1994.

[72] Economist (the), *Why the Net Should Grow Up*. The Ecomomist, pp17-18, 19 October 1996.

[73] Economist (the), *Too Cheap to Meter*. The Ecomomist, pp25-28, 19 October 1996.

[74] Eichin M.W. and Rochlis J.A, *The Microscope and Tweezers: An Analysis of the Internet Virus of November 1988*. IEEE Symposium on Research in Security and Privacy, February 1989.

[75] Ellul N. (editor), *@messages*. Internet 14, pp19-20, January 1996.

[76] Erramilli A., Narayan O. and Willinger W., *Experimental Queuing Analysis with Long-Range Dependent Packet Traffic*. IEEE/ACM Transactions on Networking. Volume 4, Number 2, pp209-223, April 1996..

[77] Fielding R., *Conditional GET Proposal for HTTP Caching*. http://www.ics.uci.edu/WebSoft/caching.html, (last-modified) 7 February 1995.

[78] Financial Times, *Faster Route to Web Sites*. Financial Times, pp17, 22 October 1997.

[79] Fisher Y., *Fractal Image Compression*, SIGGRAPH '92 Course Notes, http://inis.ucsd.edu/y/Fractals/fractal_paper.ps.Z, 1992.

[80] Fishman G.S. and Moore L.R. *A Statistical Evaluation of Multiplicative Congruential Random Number Generators with Modulus 2^{32} - 1*. Journal of the Amerian Statistical Association, Volume 77, Number 377, Theory and Methods Section, pp129-137, March 1982.

[81] Fishman G.S. and Moore L.R. *An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus 2^{32} - 1.* SIAM J. Sci. Stat. Comput, Volume 7, Number 1, Theory and Methods Section, pp24-45, January 1986.

[82] Fox B., *Wired for Sound.* New Scientist, Volume 151, Number 2049, pp25, 28 September 1996.

[83] Fox D., *Webmeister.* Internet, Issue 3, pp26-28, January 1995.

[84] Frey D. and Adams R., *A Directory of Electronic Mail Addressing and Networks.* O'Reilly & Associates, Inc., 1990. ISBN: 0-937175-175-15-3..

[85] Glance N.S. and Huberman B.A., *The Dynamics of Social Dilemmas.* Scientific American, Voume 270, Number 3, pp58-63, March 1994.

[86] Glassman S., *A Caching Relay for the World Wide Web.* Digital Equipment Corporation. http://www.research.digital.com/SRC/personal/Steve_Glassman/CachingTheWeb.html, April 1994.

[87] GNU, *RTR's WIN95Pak Man page for GZIP.* http://www.rtr.com/win95pak/gzip.htm, (last-modified) 19 March 1997.

[88] Gowar J., *Optical Communication Systems.* Prentice-Hall International Series in Optoelectronics, 1984. ISBN 0-13-638156 1.

[89] Granger C.W.J. and Joyeux R., *An Introduction to Long-Memory Time Series models and Fractional Differencing.* Journal of Time Series Analysis, Volume 1, Number 1, pp15-29, 1980.

[90] Gray J., *Cost of Messages.* Proc. 12th ACM Symposium on Principles of Distributed Computing, Toronto Canada, pp15-17, August 1988.

[91] Gray M., *Growth of the Web.* http://www.netgen.com/info/growth.html, (last-modified) September 1994.

[92] Gore A., *Infrastructure for the Global Village.* Volume 265, Number 3, Scientific American, pp108-111, September 1991.

[93] Goodman J.R., *Using Cache Memory to Reduce Processor-Memory Traffic.* Proceedings of the 10th Annual Symposium on Computer Architecture, pp124-131, June 1983.

[94] Gould J.D., Lewis C. and Barnes V., *Effects of Cursor Speed on Text-Editing.* CHI'85 conference proceedings, pp 7-9, 1985.

[95] Groeneveld P., *Digital Picture Archive on the 17th Floor,* http://olt.et.tudelft.nl/fun/pictures/pictures.html, September 1994.

[96] Gupta A., Jukic B., Parameswaran M., Stahl D.O. and Whinston A.B., *Streamlining the Digital Economy: How to Avert a Tradegy of the Commons.* IEEE Internet Computing, Volume 1, Number 6, pp38-46, November/December 1997.

[97] Gustafsson E. and Larlsson G., *A Literature Survey on Traffic Dispersion,* IEEE Network, Volume 11, Number 2, pp28-36, March/April 1997.

[98] Handel R. and Huber M.N., *Integrated Broadband Networks, An Introduction to ATM-Based Networks.* Addison-Wesley, 1991. ISBN: 0-201-54444-X.

[99] Harman G., *The Nature of Morality an Introduction to Ethics.* Oxford University Press, 1977. ISBN: 0-19-502143-6.

[100] Held G., *Digital Networking and T-Carrier Multiplexing,* John Wiley & Sons LTD, 1990. ISBN: 0-471-92800-3.

[101] Held G., *Data Compression*, John Wiley & Sons LTD, 1991. ISBN: 0-471-92941-7.

[102] Henderson H., *The USENET System*. UNIX Papers for UNIX Developers and Power Users, Howard W. Sams & Company, pp42-90, 1987. ISBN: 0-672-22578-6.

[103] Henry G.J., *The Fair Share Scheduler*. AT&T Bell Laboratories Technical Journal, Volume 63, Number 8, October 1984.

[104] Henry G.J., pers. comm., 1994.

[105] Hecht J., *Lawyers' Ad Challenges Rules of 'Netiquette'*. New Scientist, Volume 143, Number 1933, pp19, 9 July 1994.

[106] Hix D. and Hartson H.R., *Developing User Inetrfaces*. John Wiley & Sons, Inc. 1993. ISBN: 0-471-57813-4.

[107] Holt A.G., *Do Disk Drives Dream of Buffer Cache Hits*? Proceedings of the ACM Ethics in the Computer Age Conference, November 1994.

[108] Holt A.G. *The Network Application from Hell*. Social and Ethical Effects of the Computer Revolution, pp162-170, McFarland and Company Inc., 1996.

[109] Holt A.G. *Prisoners, Chicken, Volunteers, Free Riders and Suckers: Dilemmas on the Internet*. IEE Engineering Science and Education Journal, Volume 6, pp73-77, April 1997.

[110] Honderich, T. (editor), *The Oxford Companion to Philosophy*. Oxford University Press, 1995. ISBN: 0-19-866132-0.

[111] Hosking J.R.M., *Fractional Differencing*. Biometrica, Volume 68, Number 1, pp165-176, 1981.

[112] Hsieh M.M., Wei T.C. and Van Loo W., *A Cached System Architecture Dedicated for the System IO Activity on a CPU Board.* Sun Microsystems, International Conference on Computer Design, 10 February 1989.

[113] Hughes K., *How Popular is the Web?* Enterprise Integration Technologies, http://www.eit.com/web/www.guide/guide.05.html, May 1994.

[114] Hughes K., *Guide to Cyberspace.* Enterprise Integration Technologies, URL http://www.eit.com/web/www.guide/, Version 6.1.1, March 1995.

[115] Huitema C., *Routing in the Internet.* Prentice Hall, 1995. ISBN: 0-13-132-192-7.

[116] Hull T.E. and Dobell A.R., *Random Number Generators.* SIAM Review, Volume 4, Number 3, pp230-254, July 1962.

[117] Husselbaugh B., *(Mis)using Bandwidth.* Byte, Volume 19, Number 12, pp117-124, December 1994.

[118] Hutchinson D.W., *A New Uniform Psuedorandom Number Generator.* Commuications of the ACM, Volume 9, Number 6, pp432-434, June 1966.

[119] id Software, *Doom II Instruction Manual.* id Software Inc., 1994.

[120] Intel *Q&A Forum, Intel Corporation,* http://www.intel.com/businesscomputing/tech/work/q_and_a/index.htm, 1998.

[121] Isaac R., *The Pleasure of Probability.* Springer-Verlag Inc., 1995. ISBN: 0-387-94415-X.

[122] Jacobson V., *Congestion Avoidance and Control.* Computer Communications Review, Volume 18, Number 4, pp 314-329, Proceedings of the ACM SIGCOMM '88 Workshop, August 1988.

[123] Jain R., *The Art of Computer System Performance Analysis.* John Wiley & Sons, Inc. 1991, ISBN: 0-471-50336-3.

[124] Jamieson D., *Ethics, Public Policy, and Global Warming.* Applied Ethics, Blackwell, edited by Winkler E.R. and Coombs J.R. pp313-328, ISBN: 0-631-18832-0.

[125] Jannife P., *Canter & Siegel Spam Wars.* On-line World, pp36-40, November 1994.

[126] Johnson T.V., *The Galileo Mission.* Scientific American, Volume 276, Number 1, pp28-35, December 1995.

[127] Kac M., *What is Random?* American Scientist, Volume 71, pp405-407, July-August 1983.

[128] Kapor M., *Civil Liberties in Cyberspace.* Scientific American, Volume 265, Number 3, pp116-120, September 1991.

[129] Kay D.C and Levine J.R, *Graphics File Formats.* Windcrest/McGraw-Hill, second edition, 1995. ISBN: 0-07-034025-0.

[130] Kendall M. and Ord J.K., *Time Series.* Edward Arnold, Third edition, 1990, ISBN: 0-340-59327-X.

[131] Kernighan B.W. and Pike R.P., *The Unix Programming Environment.* Prentice Hall, Inc Englewood Cliffs, 1984. ISBN: 0-13-937681-X.

[132] Kernighan B.W. and Ritchie D.M., *The C Programming Language.* Prentice Hall, Inc Englewood Cliffs, Second edition 1988. ISBN: 0-13-1103262-8.

[133] Kennedy J., *Mosaic.* Internet and Comms Today, Issue 3, pp46-49, January 1995.

[134] Knight J. and Guest S., *Using Multicast Communications to Distribute Code and Data in Wide Area Networks.* Software Practise and Experience, Volume 25(5), pp563-577, May 1995.

[135] Knuth D.E., *Fundamental Algorithms: The Art of Computer Programming.* Addison-Wesley Publishing Company, second edition, 1973. ISBN: 0-201-03809-9.

[136] Kleiner K., *Wanna Buy a Computer Time-share?* New Scientist, Volume 151, Number 2045, pp20, 31 August 1996.

[137] Kleinrock L., *Queuing Systems Volume I: Theory.* Wiley and Sons, 1975. ISBN: 0-471-49110-1.

[138] Kroll E., *The Whole Internet User Guide & Catalog.* O'Reilly & Associates, Inc., 1993. ISBN: 1-56592-025-2.

[139] Kwan T., McGrath R. and Reed D. *User Access Patterns to NCSA's World Wide Weeb Server.* http://www-pablo.cs.uiuc.edu/Papers/WWW.ps.Z, 17 February 1995.

[140] Lane T., *JPEG Image Compression: Frequently Asked Questions*, http://www.cis.ohio-state.edu/hypertext/faq/usenet/jpeg-faq/faq.html, (last-modified) August 1995.

[141] LaQuay T.L., *The User's DIRECTORY Computer Networks.* Digital Press/Prentice Hall International Edition, 1990. ISBN: 0-13-950759-0.

[142] Lavin P., *Attitude Problem.* Internet, Issue 5, pp24, April 1995.

[143] L'Ecuyer, *Efficient and Portable Combined Random Number Generators.* Communications of the ACM, Volume 31, pp742-774, 1988.

[144] Leffler S.J., McKusik M.K., Karels M.J. and Quarterman J.S., *The Design and Implementation of the 4.3BSD Unix Operating System.* Addison-Wesley Publishing Company, 1990.

[145] Leong H.V. and Si A. *Database Caching Over the Air-storage.* the Computer Journal, Volume 40, Number 7, pp401-415; 1997.

[146] Li W.K., *Fractional Time Series Modelling.* Biometrika, Volume 73, Number 1, pp2217-221, 1988.

[147] Lippis N. and Morency J., *Reducing Router Network Transmission Costs.* Byte, pp70-71, Volume 18, Number 3, December 1993.

[148] Liu P., *LAN Traffic Symptoms & Problem Solving Case Studies from LAN Traffic Monitoring.* AT&T Bell Laboratories Presentation.

[149] Liu P.C., *Preliminary Assessment of Wide-area File Sharing.* AT&T Technical Memorandum 45262-910801-01IM, August 1991.

[150] Loukides M., *System Performance Tuning.* O'Reilly & Associates, Inc., 1991.

[151] Luotonen A. and Altis K., *World-Wide Web Prog.* http://www.city.net/cnx/kevin_altis/papers/Proxies/, 24 May 1994.

[152] Lynch C., *Searching the Internet.* Scientific American, Volume 276, Number 1, pp44-48, March 1997.

[153] Lynch M., *Push and Pull is Dead.* Internet Business, Issue 13, pp58-59, February1998.

[154] MacKie-Mason J.K. and Varian H.R., *Pricing the Internet.* Public Access to the Internet, pp269-314, The MIT Press, 1995. ISBN: 0-262-11207-8.

[155] Malone T.W. and Rockart J.F., *Computer Networks and the Corporation.* Scientific American, Volume 265, Number 3, pp92-99, September 1991.

[156] Manley J.E. and Trieber S.P., *Ethernet Bandwidth Utilisation of Datafull, Dataless, and Diskless Node Workstations and X-Terminals*. The MITRE Corporation, Bedford, MA, November 1991.

[157] Markatos E.P., *Main Memory Caching of Web Documents*. Computer Networks and ISDN Systems, Volume 28, pp893-905, 1996.

[158] Marsaglia G., *Random Numbers Fall Mainly in the Planes*. Acad. Sci. Proc., Volume 61, pp25-28, September 1968.

[159] Marsaglia G. and Bray T.A., *One-Line Random Number Generators and Their Use in Combinations*. Communications of the ACM., Volume 11, Number 11, pp757-759, November 1968.

[160] Mathews G.J., *Evaluating Data-Compression Algorithms*. Dr Dobbs Journal, pp50-53, January 1996.

[161] Mathews J., *Video Compression Techniques*. Electronics World , pp102-108, February 1996.

[162] McArthur D. C., *World Wide Web & HTML*. Dr Dobb's Journal, pp18-26, December 1994.

[163] McCormick B., DeFanti T and Brown R. (Editors), *Visualisation in Scientific Computing and Computer Graphics*. ACM SIGGRAPH 21, November 1987.

[164] McGregor D.R., Fryer R.J., Cockcroft P. and Murray P., *Faster Fractal Compression*. Dr Dobbs Journal, pp34-40 , January 1996.

[165] McKnight L.W. and Bailey J.P., *Internet Economics: When Constituencies Collide in Cyberspace*. IEEE Internet Computing, Volume 1, Number 6, pp30-37, November/December 1997.

[166] McLean F., *Integrating Ethics and Design*. IEEE Technology and Society Magazine, Fall 1993.

[167] Miller L.H., *A Study in Man-Machine Interaction*. National Computer Computer Conference, pp 409-421, 1977.

[168] Minoli D., *Broadband Network Analysis and Design*. Artech House Inc.,1993. ISBN: 0-89006-675-2.

[169] Mokhtarian P.L., *Now That Travel Can be Virtual, Will Congestion Virtually Disappear?* http://www.sciam.com/1097issue/1097mokhtarian.html, October 1997.

[170] Molle M., Kalkunte M. and Kadambi J., *Frame Bursting: A Technique for Scaling CSMA/CD to Gigabit Speeds*. IEEE Network, Volume 11, Number 4, pp6-15, July/Aaugyst 1997.

[171] Moody G. *Internet - Now Available on Disc*. The Guardian, pp7, 30 November 1995.

[172] Moore G.E., *Principia Ethica*. The Cambridge University Press 1959.

[173] Morris P., *Introduction to Game Theory*. Springer-Verlag, 1994. ISBN 0-387-94284-X.

[174] Morningstar C. and Farmer F.R., *Lessons of LucasFilm Habitat*. Cyberspace First Steps, pp 273-300, The MIT Press 1991, ISBN: 0-262-02327-X.

[175] Mosedale D., William F. and McCool R. *Lessons Learned Administering Netscape's Internet Site*. IEEE Internet Computing, Volume 1, Number 2, March/April 1997.

[176] Mullender S.J. and Tanenbaum A.S., *Immediate Files*. Software - Practise and Experience, Volume 14(4), pp365-368, April 1984.

[177] Muntz D. nad Honeyman P. *Multi-Level Caching in Distributed File Systems - or - Your Cache Ain't Nuthin' but Trash"*, Proc. 7th IEEE Sump. Parallel and Distributed Processing, October 1995.

[178] NASA, *Latest Galileo FAQ Questions*. http://www.jpl.nasa.gov/galileo/newfaq1.html#compression, 14th September, 1995.

[179] NASA, *New Telecommunications Strategy*. http://www.jpl.nasa.gov/galileo/, 1 November, 1995.

[180] Negrponte N.P., *Products, Services and Computer Networks*. Volume 265, Number 3, Scientific American, pp76-83 , September 1991.

[181] Negroponte N., *Being Digital*, Hodder & Stoughton, 1995. ISBN 0-340-64525-3.

[182] Negroponte N., *Affordable Computing*, Wired, pp114, July/August 1995.

[183] Netscape, *Caching Parameters*. Netscape Communications Corporation, http://mlweb.mlm.att.com:8000/admin/cacheparams.htm, (last-modified) 7th March 1995.

[184] Netscape, *Netscape Handbook: Menu Items*. http://www.netscape.com/newsref/manual/docs/menu.html#C15, (last-modified) 15th April 1995.

[185] Netscape, *Netscape Handbook: Answers to Tough Questions*. http://www.netscape.com/newsref/manual/docs/answers.html#C8, (last-modified) 15th April 1995.

[186] Netscape, *Netscape Proxy Server Data Sheet*. http://home.netscape.com/comprod/server_central/product/proxy/proxy_data.html, (last-modified) 14 October 1996.

[187] Network Week, *Broadband Costs Keep Speeds Low*, pp1, 30th July 1997.

[188] Nielsen J., *Zipf Curves and Website Popularity.* http://www.useit.com/alertbox/zipf.html, April 1997.

[189] Park S.K. and Miller K.W., *Random Number Generators: Good Ones are Hard to Find.* Communications of the ACM, Volume 31, Number 10, pp1192-1201, October 1988.

[190] Partridge C., *How Slow Is One Gigabit Per Second?* ACM Computer Communication Review, Volume 20, Number 1, pp 44-53, January 1990.

[191] Partridge C., *Gigabit Networking.* Addison-Wesley Professional Computing Series, 1994. ISBN: 0-201-56333-9.

[192] Partridge C., Carvey P.P., Burgess E., Castineyra I., Clarke T., Graham L., Hathaway M., Herman P., King A., Kohalmi S, Ma T., Mcallen J., Mendez T., Milliken W.C., Pettyjohn R., Rokosz T., Seeger J, Sollins M., Storch S., Tober B., Troxel G.D. Watzman D. and Winterble S., *A 50-Gb/s IP Router,* IEEE/ACM Transaction on Networking, Volume 6, Number 3, pp237-248, June 1998.

[193] Payne W.H., Rabung J.R. and Bogyo T.P. *Coding the Lehmer Pseudorandom Number Generator.* Commuications of the ACM, Volume 12, Number 2, pp85-86, February 1969.

[194] Paxson V., *Measurements and Models of Wide Area TCP Conversations.* Computer Systems Engineering Department Lawrence Berkeley Laboratory, DoE Contract Number DE-AC03-76F00098. May 1991.

[195] Penny J.P., Ashton P.J. and Wilkinson A.L., *Data Recording and Monitoring for Analysis of System Response Times.* The Computer Journal, Volume 29, Number 5, 1986.

[196] Peitgen H. and Saupe D. (editors), *The Science of Fractal Images.* Springer-Verlag, 1988. ISBN: 0-387-96608-0.

[197] Peitgen H., Jurgend H., and Saupe D., *Fractals for the Classroom.* Part 1, Springer-Verlag, 1992. ISBN: 0-387-97041-X.

[198] Press W.H., Teukolsky S.A., Vetterling W.T. and Flannery B.P., *Numerical Recipes in C.* Cambridge University Press, Second Edition, 1992. ISBN 0-521-43108-5.

[199] Port O., *Through a Glass Quickly.* Business Week, pp96-98, 7December 1998.

[200] Poundstone W., *Prisoner's Dilemma.* Oxford University Press, 1992, ISBN: 0-19-286162-X.

[201] dePryker M., *Asynchronous Transfer Mode Solution for Broadband ISDN.* Ellis Horward, 1991. ISBN 0-13-053513-3.

[202] Ritchie D.M., *Unix Time-Sharing System: A Retrospective.* The Bell System Technical Journal, Volume 57, Number 6, July-August 1978.

[203] Rago S.A., *Unix Systems V Network Programming.* Addison-Wesley Professional Computing Series, 1993. ISBN: 0-201-56318-5.

[204] Reid E.M., *Electropolis: Communications and Community on Internet Relay Chat.* Honours Thesis (emr@ariel.ucs.unimelb.edu.au), 1991.

[205] Rennie J. et al (editors), *The Internet: Bringing Order to Chaos.* Scientific American, Volume 265, Number 3, pp42-43, March 1997.

[206] Rheingold H., *Virtual Reality.* Secker & Warburg, 1991. ISBN: 0-436-41212-8.

[207] Rillings J.H., *Automated Highways.* http://www.sciam.com/1097issue/1097rillings.html, October 1997.

[208] Sach L., *Applied Statistics*, A Handbook of Techniques. Springer-Verlag, 1984. ISBN: 0-387-90976-1.

[209] Salem L., Terard F. and Salem C., *The Most Beautiful Mathematical Formulas.* John Wiley & Sons, Inc., 1992. ISBN: 0-471-55276-3.

[210] Santifaller M. *TCP/IP and NFS Internetworking in a Unix Environment.* Addison-Wesley Publishing Company, 1991. ISBN: 0-201-54432-6.

[211] Schimmel C., *Unix Systems for Modern Architectures.* Addison-Wesley Professional Computing Series, 1994. ISBN: 0-201-63338-8.

[212] Schneider T.D., *Information Theory Primer,* Version 2.32, ftp://ftp.ncifcrf.gov/pub/delila/primer.ps, July 1995.

[213] Shannon C.E. and Warren W., *A Mathematical Theory of Communication.* University of Illinois Press, 1949. ISBN: 0-252-72548-4

[214] Shoch J.F. and Hupp J.A., *Measured Performance of an Ethernet Local Network.* Communications of the ACM Volume 23, Number 12, December 1980.

[215] Shoch J.F. and Hupp J.A., *Performance of an Ethernet: a Preliminary Report.* Proceedings of the LACN Symposium, May 1979.

[216] Shneiderman B., *Designing the User Interface.* Addison-Wesley Publishing Company Inc., Second Edition, 1992. ISBN:0-201057286-9.

[217] Smith A.J., *Cache Memories.* Computing Surveys, Volume 14, Number 3, pp 473-530, September 1993.

[218] Smith N.G., *The UK National Web Cache - The State of the Art.* Computer Networks and ISDN Systems, Volume 28, pp 1407-1414, 1996.

[219] Song C. and Landweber L.H., *Optimizing Bulk Data Transfer Performance: A Packet Train Approach.* ACM 0-897910279-9/88/008/0134 pp134-145, 1988.

[220] Stallings W., *The Open Systems (OSI) Model and OSI-Related Standards. Handbook of Computer Communications Standards.* Volume 1, Second Edition, Howard W. Sams & Company, 1990. ISBN: 0-672-22697-9.

[221] Stallings W., *Local Area Networks Standards. Handbook of Computer Communications Standards.* Volume 2, Second Edition, Howard W. Sams & Company, 1990. ISBN: 0-672-22698-7

[222] Stallings W., *TCP/IP Protocol Suite. Handbook of Computer Communications Standards.* Volume 3, Second Edition, Howard W. Sams & Company, 1989. ISBN: 0-672-22696-0.

[223] Statistical Sciences, *S-Plus Guide to Statistical and Mathematical Analysis.* Version 3.2, Seattle: StatsSci. a division of MathSoft, Inc., 1993.

[224] Stevens W.R., *Unix Network Programming.* Prentice Hall Software Series, 1990. ISBN: 0-13-949876-1.

[225] Stevens W.R., *Advanced Programming in the Unix Environment.* Addison-Wesley, June 1992. ISBN: 0-201-56317-7.

[226] Stix G., *Encoding the "Neatness" of Ones and Zeroes.* Scientific American, Volume 269, Number 2, pp 27-28, September 1991.

[227] Stix G., *Domesticating Cyberspace.* Scientific American. Volume 269, Number 2, pp84-92, August 1993.

[228] Stoll C., *The Cuckoo's Egg.* The Bodley Head, 1989. ISBN:0-370-31433-6.

[229] Tanenbuam A.S., *Computer Networks.* Englewood Cliffs, NJ: Prentice Hall, ISBN: 09130162959-X, 1988.

[230] Tanenbuam A.S., *Modern Operating Systems.* Prentice Hall International Editions, ISBN: 0-13-595752-4, 1992.

[231] Taylor J., *Engineering the Information Age*. IEE Review, Volume 44, Number 6, pp250-252, 19 November 1998.

[232] Tennant D., *Netiquette*. Newsweek, 18 March 1996.

[233] Tesler L.G., *Networked Computing in the 1990s*. Scientific American, Volume 265, Number 3, pp54-61, September 1991.

[234] Thompson K., Miller G.J. and Wilder R., *Wide-Area Internet Traffic Patterns and Characteristics*. IEEE Network, Volume 11, Number 6, November/December 1997.

[235] Tombaugh M.D., Arkin D. and Dillon R., *The Effect of VDU Text-Processing Rate on Reading Comprehension and Reading Speed*. CHI'85 conference proceedings, pp 1-6, 1985.

[236] Treese W., *Solutions to Internet Traffic Jams*. Byte, Volume 19, Number 11, pp 27, November 1994.

[237] Trewitt G., *Local Area Internetworks: Measurements and Analysis*. NSL Research Report RR-1, Digital Network Systems Laboratory, NSL-Techreports@pa.dec.com, March 1990.

[238] Vandre Wiel S.P. and Lilja D.J., *When Caches Aren't Enough: Data Prefetching Techniques*. Computer, Volume 30, Number 7, July 1997, pp23-30, July 1997.

[239] Viles C.L. and French J.C., *Availability and Latency of World Wide Web Information Server*. Computer Systems, Volume 8, Number 1, pp61- 91, Winter 1995.

[240] Voldman J., Mandelbrot B., Hoevel L.W., Knight J. and Rosenfeld P., *Fractal Nature of Software-Cache Interaction*. IBM J. Res. Develop, Volume 27, Number 2, pp164-170, March 1983.

[241] Walsh R.J., *DART: Fast Application-Level Networking via Data-Copy Avoidance.* IEEE Network, Volume 11, Number 4, pp28-38, July/August 1997.

[242] Wang Z., and Crowcroft J., *Prefetching in the World Wide Web.* Proceedings IEEE Global Internet, pp28-32, 1996.

[243] Wayt W., *Bandwith Unlimited.* Scientific American, Volume 276, Number 1, pp30, January 1997.

[244] Wayt W., *Beyond Modems.* http://www.sciam.com/explorations/022497connect/022497gibbs.html, February 1997

[245] Willinger W., Taqu M.S., Sherman R. and Daniel V.W., *Self-Similarity High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level.* . IEEE/ACM Transactions on Networking. Volume 5, Number 1, pp71-86, February 1997.

[246] Winner L., *Citizen Virtues in a Technological Order.* Applied Ethics: A Reader, Blackwell, pp46-68, 1993.

[247] Wolfram S., *Mathematica.* Addison-Wesley, second edition, 1991.

[248] Zipf G.K., *The Psycho-Biology of Language.* Houghton Mifflin Company (Boston), 1935.

Appendix A

The graphs in this appendix show the results of the compression performance experiments from CHAPTER 3 for each individual file in the Calgary Corpus Suite.

Figure 82 paper1

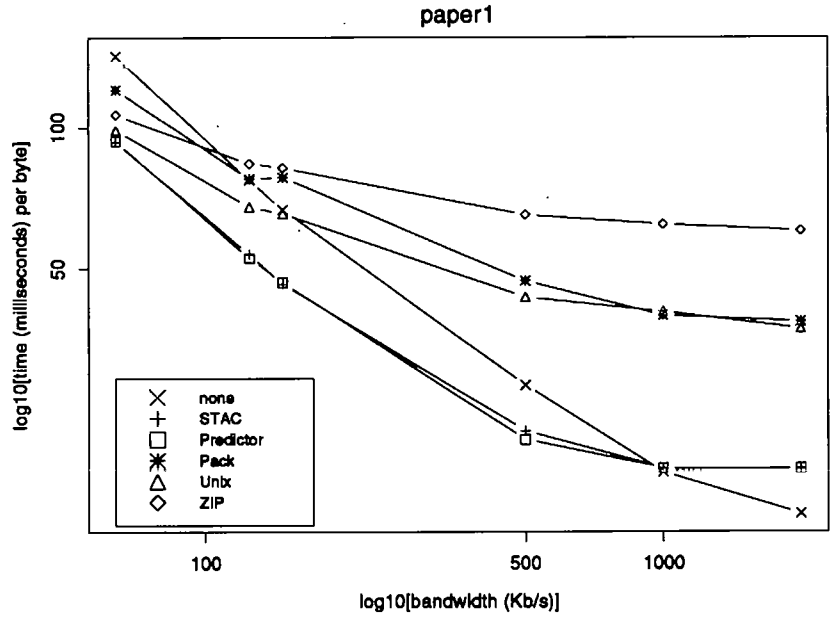


Figure 83 paper2

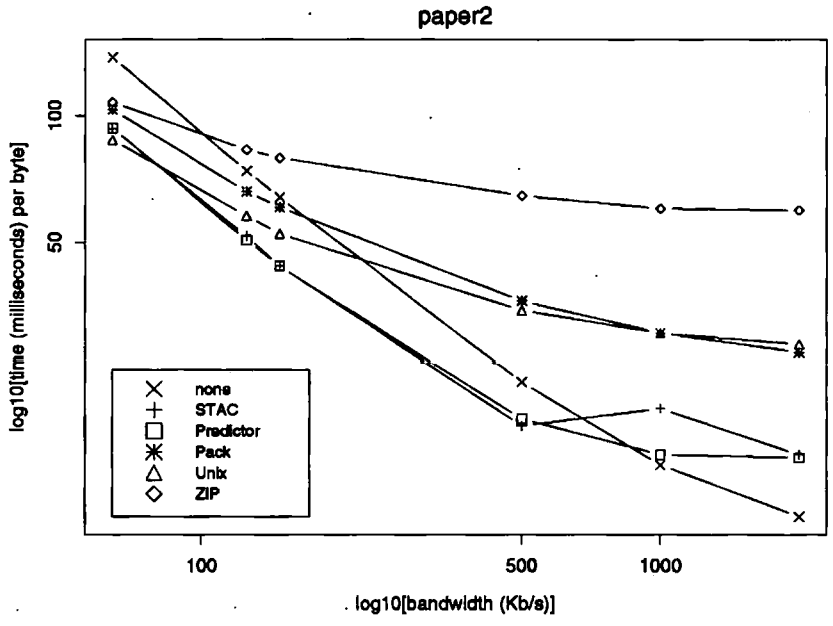


Figure 84 paper3

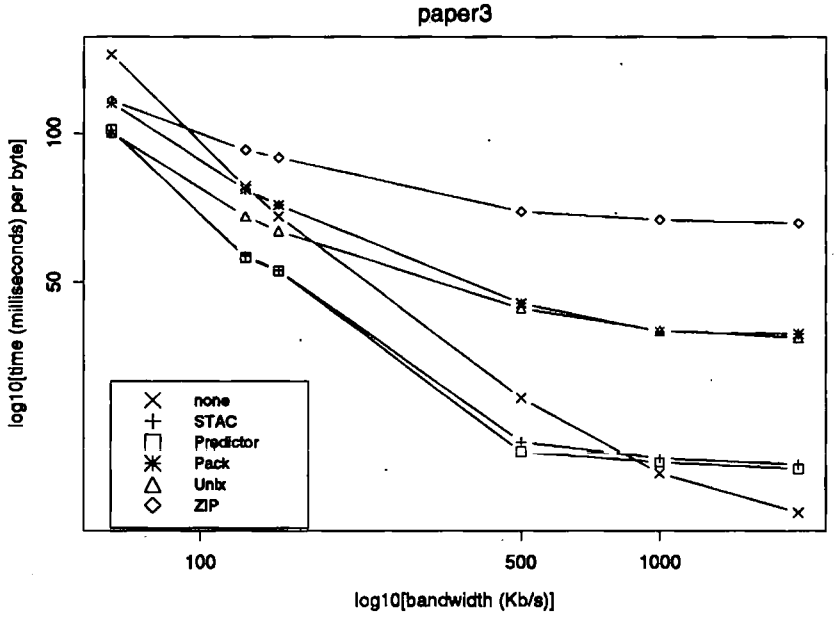


Figure 85 paper4

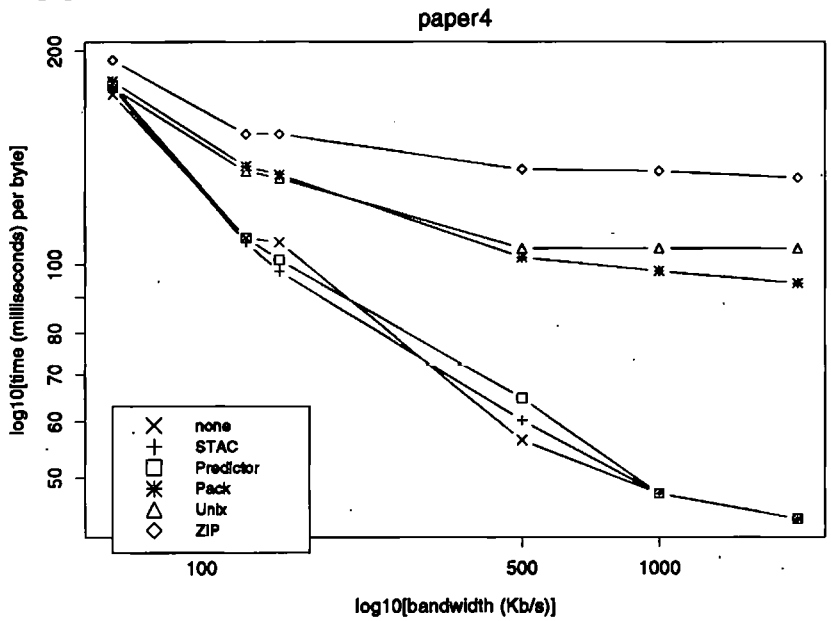


Figure 86 paper5

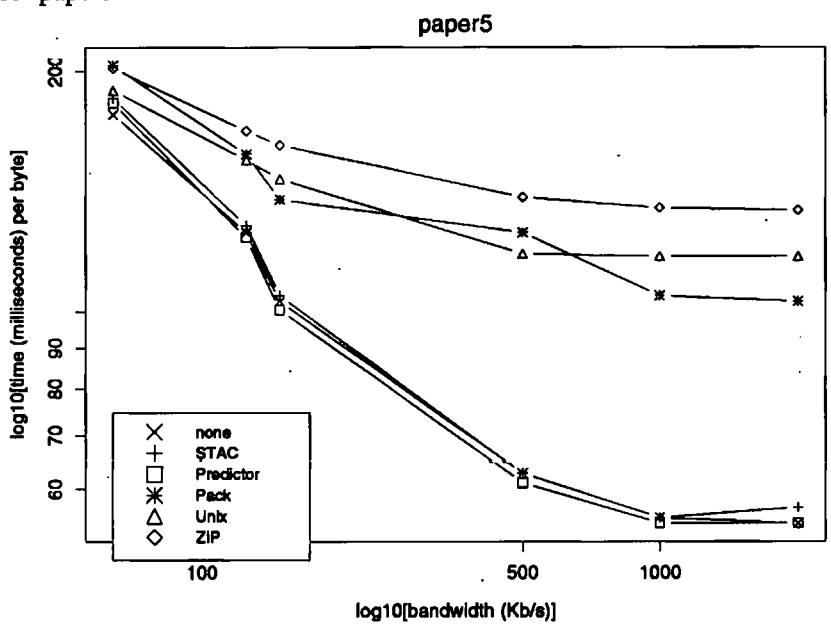


Figure 87 paper6

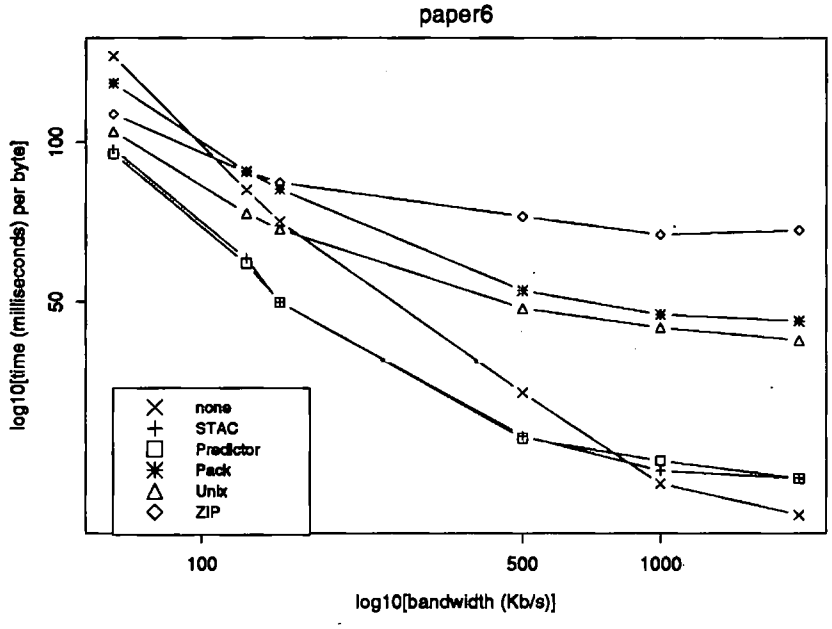


Figure 88 book1

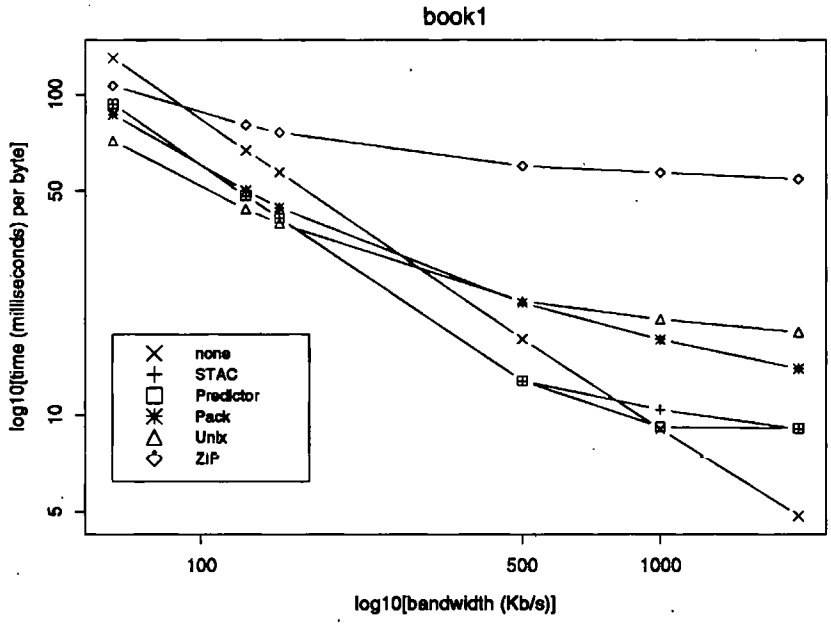


Figure 89 book2

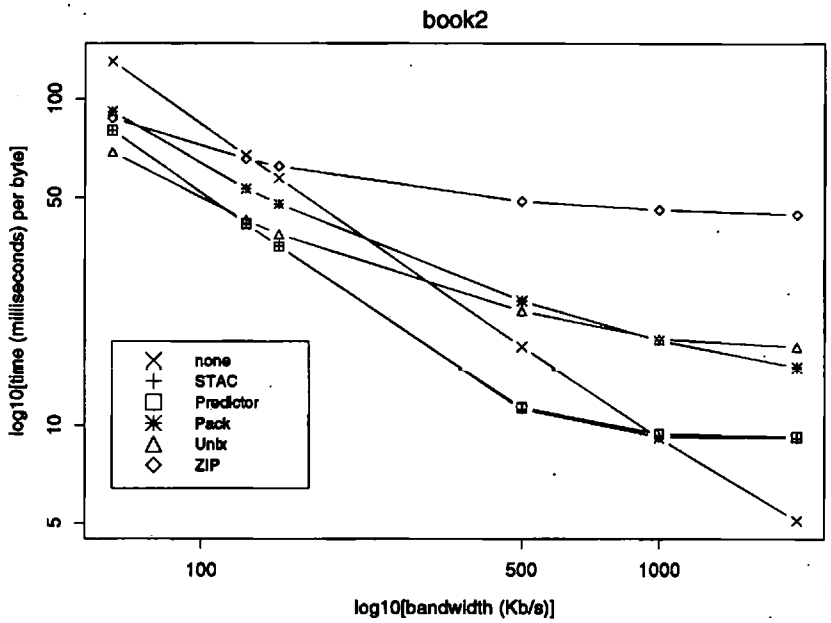


Figure 90 bib

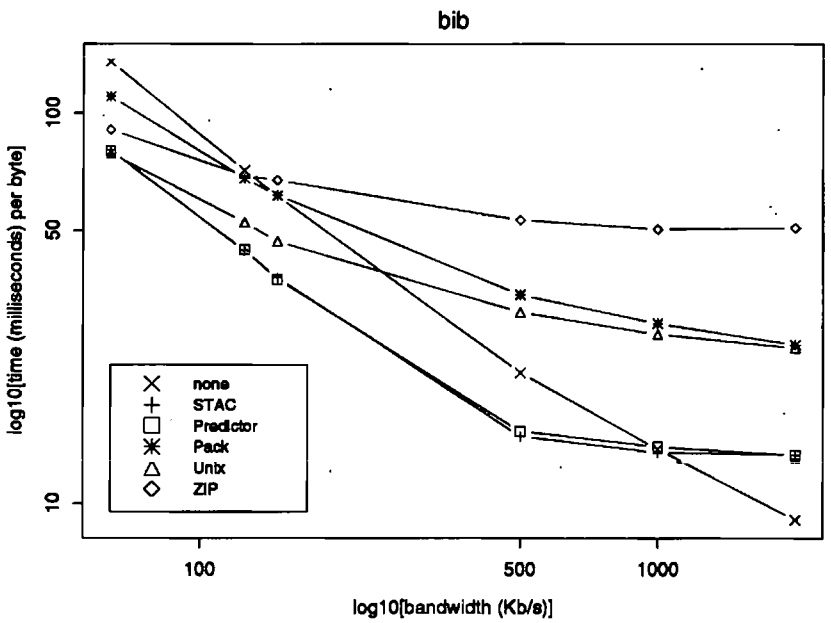


Figure 91 geo

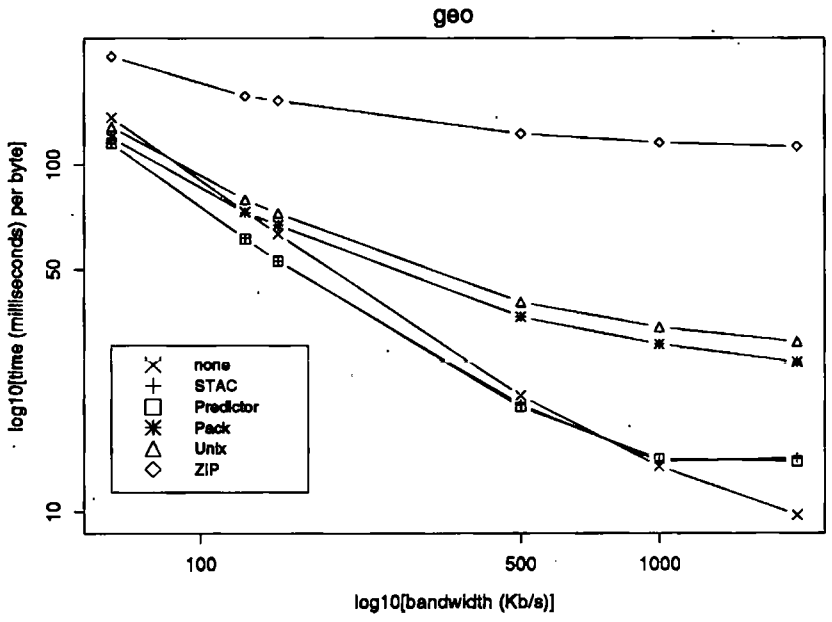


Figure 92 news

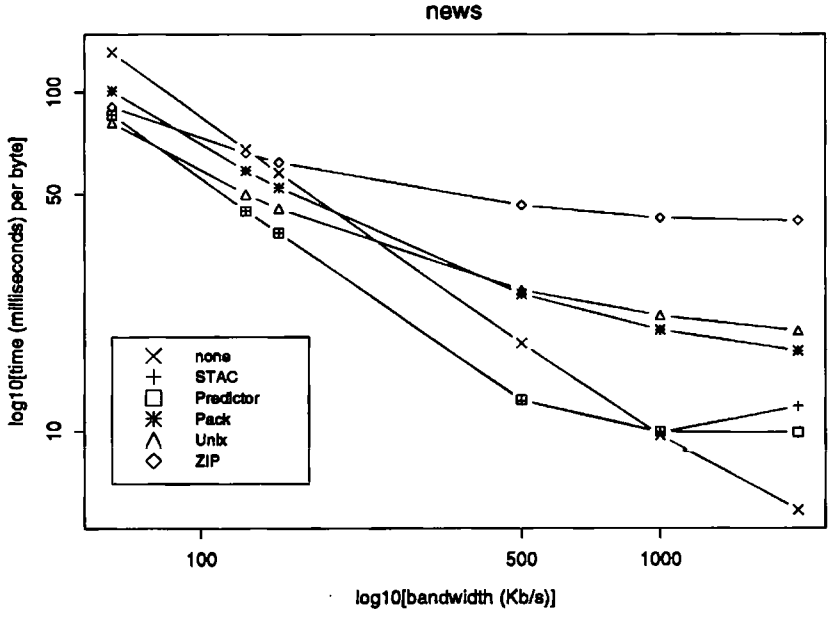


Figure 93 pic

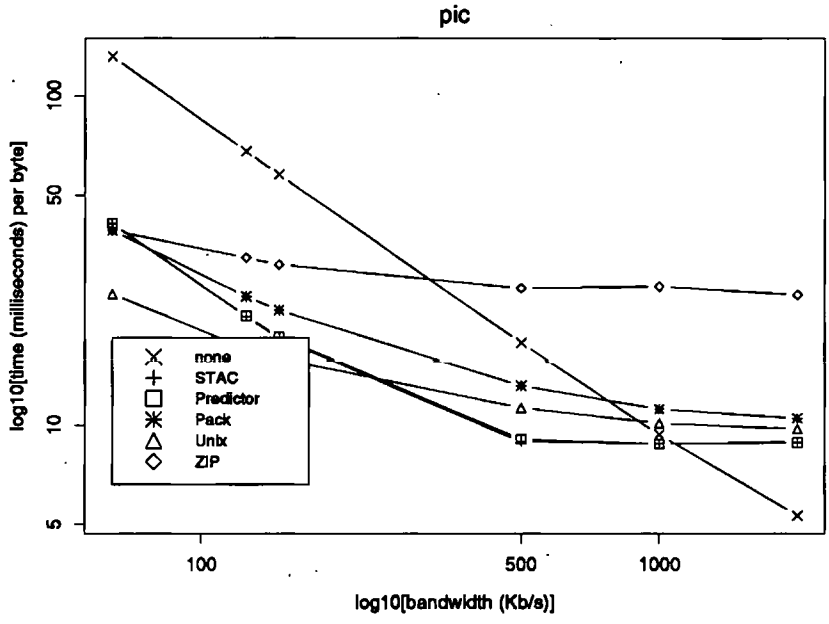


Figure 94 obj1

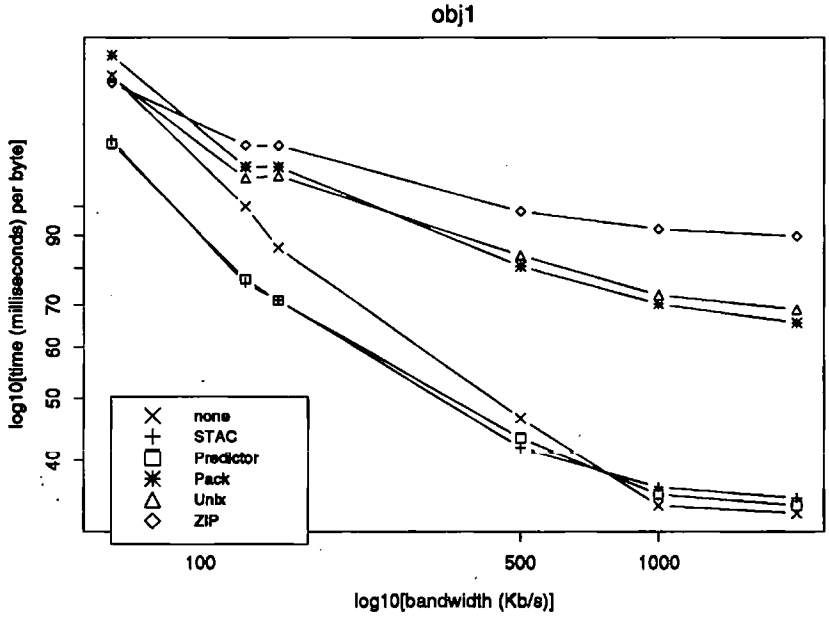


Figure 95 obj2

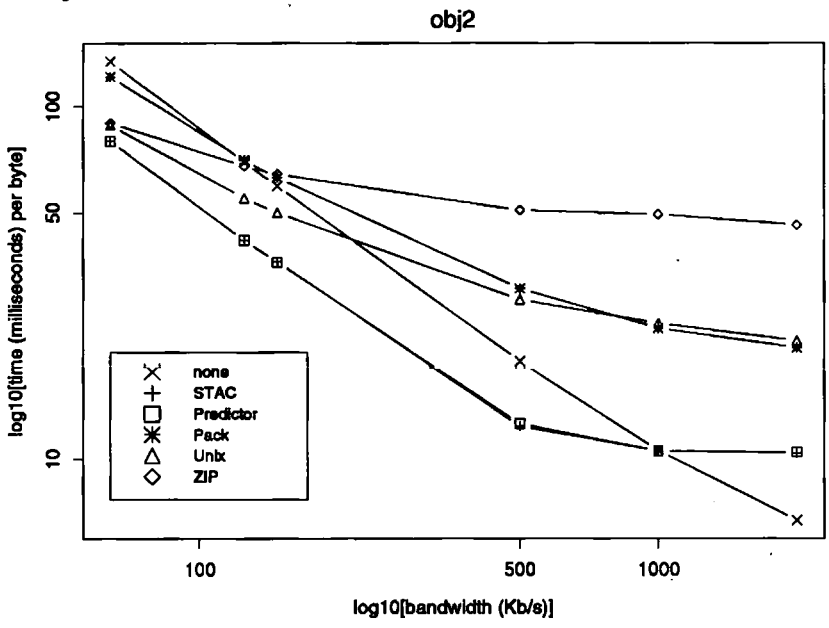


Figure 96 progc

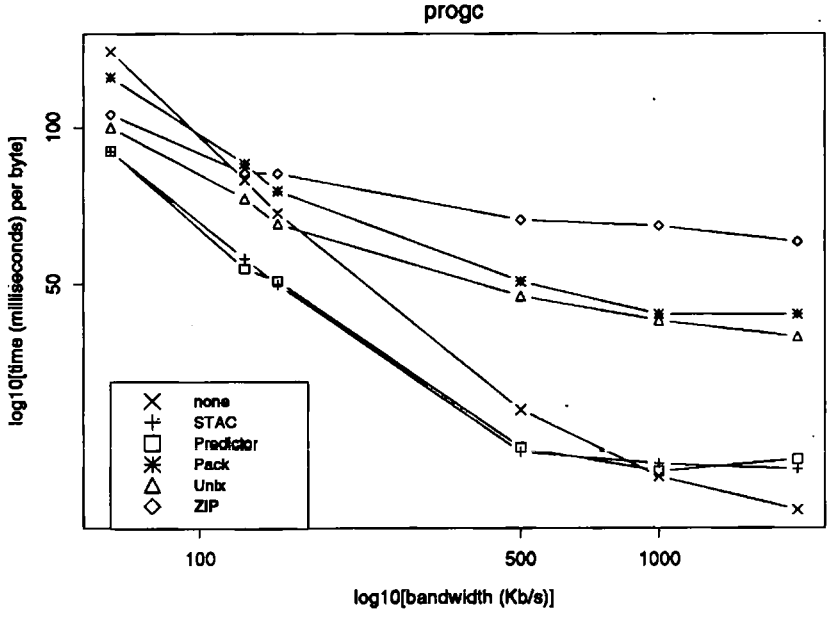


Figure 97 progl

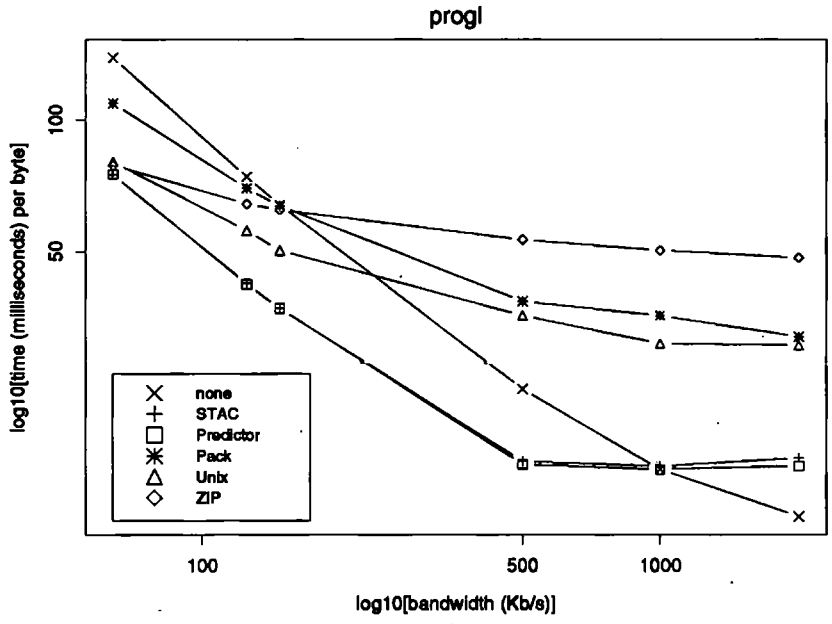


Figure 98 progp

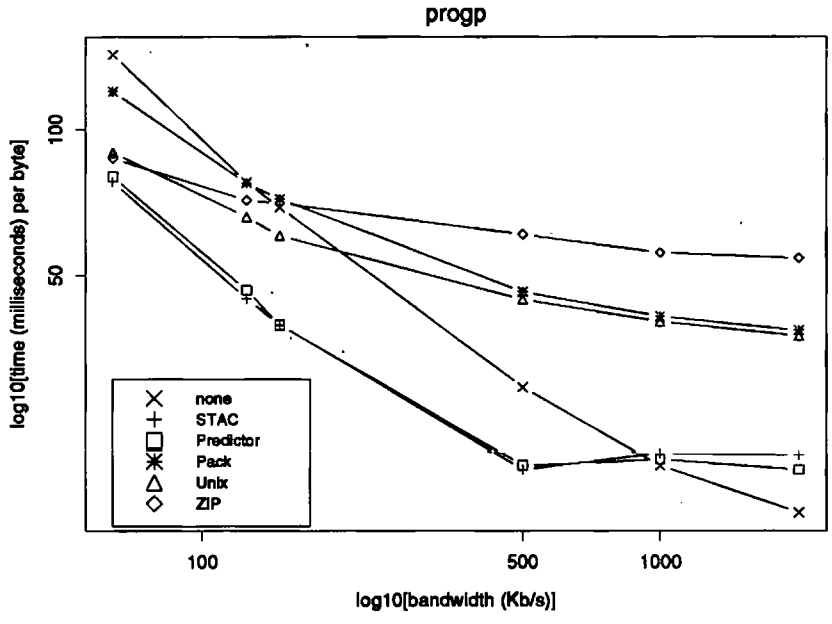
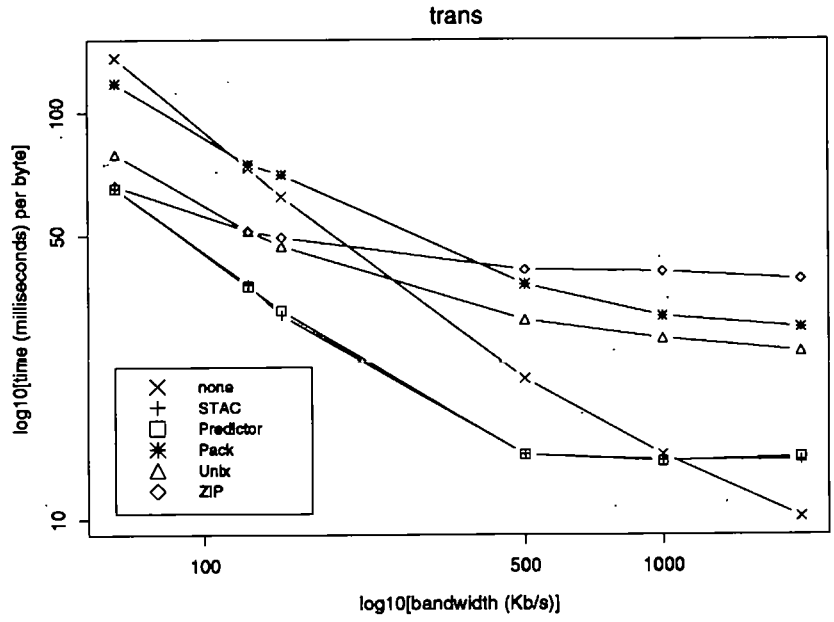


Figure 99 trans



Appendix B

In CHAPTER 5 Monte Carlo Simulation techniques were used to model the growth of cache occupancy. Cache references were modelled on Fractional autoregressive-integrated moving average processes. Recall that the ARIMA(0,*d*,0) process is given by

$$\nabla^d x(t) = \varepsilon_t \tag{83}$$

where ε_t is an independent Gaussian random variable. In this Appendix a number of methods of generating Gaussian random variates are reviewed. All methods reviewed here require the generation of uniform random variates, so a number of uniform random number generators are examined for statistical compliance, period, performance and portability.

Gaussian Random Number Generators

There are a number of method for generating Gaussian Random numbers. Three methods are examined

- Convolution
- Polar
- Rejection

The Convolution method works on the principle that the sum of a large number of uniform random variates approaches a standard normal distribution (Peitgen et. al. [197] provide a description of the Convolution method) for n variates

$$\varepsilon_t = \frac{1}{A}\sqrt{\frac{12}{n}}\sum_{i=1}^{i=n} u_i - \sqrt{3n} \tag{84}$$

where $0 \le u_n < 1$ is a uniformly distributed random variate, A is...

For the polar method generate

$$v_1 = 2u_1 - 1 \tag{85}$$

and

$$v_2 = 2u_2 - 1 \tag{86}$$

where $0 \le u_1 < 1$ and $0 \le u_2 < 1$ are uniformerly distributed random variates. Compute $r = v_1^2 + v_2^2$ and if $r \ge 1$ regenerate v_1 and v_2. Otherwise, let $s = \sqrt{(-2\ln r)/r}$ and compute two Gaussian random variables

$$\varepsilon_1 = \mu + \sigma v_1 s \tag{87}$$

and

$$\varepsilon_2 = \mu + \sigma v_2 s \tag{88}$$

with mean μ and standard deviation σ. The code for a version of the Polar method was taken from Press et. al. [198].

The Rejection Method requires the generation of two uniformerly distributed random variates $0 \le u_1 < 1$ and $0 \le u_2 < 1$. Calculate $x = -\ln u_1$ and if $u_2 > e^{-(x-1)^2/2}$ regenerate u_1 and u_2. Otherwise return a Gaussian random variate

$$\varepsilon_t = \begin{cases} \mu + \sigma x & \text{with probability} \quad p = 0.5 \\ \mu - \sigma x & \text{with probability } q = 1 - p \end{cases} \tag{89}$$

with mean μ and standard deviation σ.

Uniform Random Number Generators

It can be seen that all of these methods require uniform random numbers. Almost all uniform random number generators are *linear congruential generators* [198]. Linear congruential generators are of the form

$$I_{j+1} = aI_j + c \qquad (\text{mod } m) \tag{90}$$

and produce sequences of number between 0 and $m - 1$. Four linear congruential random number generators were examined. These are listed in Table 13.

Table 13 Random number generators

Generator	Source
`ran0`	Park and Miller [189]/Press et. al. [198]
`ran2`	L'Ecuyer [143]/Press et. al. [198]
`ranqd2`	Press et. al. [198]
`erand48`	Posix IEEE [229]

The criteria by which these generators were assessed were:

- Statistical compliance
- Period
- Performance
- Portability

Sequences of "random numbers" generated by a computer algorithm are not strictly random and are often referred to as *pseudo-random*. Nevertheless pseudo-random

sequences (provided they are generated by a good random number generator) can pass statistical tests for randomness and distribution.

One hundred consecutive replications of 100,000 variates from each generator were subjected to the following statistical tests

- Chi-square
- Kolmogorov-Smirnov
- Serial-correlation

Statistical Compliance

The Chi-square test is used to determine if the variates from each generator are uniformly distributed

$$\chi^2 = \frac{k}{n}\sum_{i=1}^{i=n}\left(f_i - \frac{n}{k}\right)^2 \tag{91}$$

Using a level of significance of $\alpha = 0.001$, $\chi^2_{0.99,99,999} = 63.7$. As none of the replications from any generator exceeded this value the random numbers generated were accepted as being uniformly distributed.

Similarly the Kolmogorov-Smirnov test was used to test for uniformity

$$K = \max\left|S_N(x) - P(x)\right| \qquad -\infty < x < \infty \tag{92}$$

For each generator there were no observed values of K that exceeded $K_{[0.99;100000]} = 1.48$ in 100 replications. So random variates generated were accepted as being uniformly distributed for a level of significance of $\alpha = 0.001$

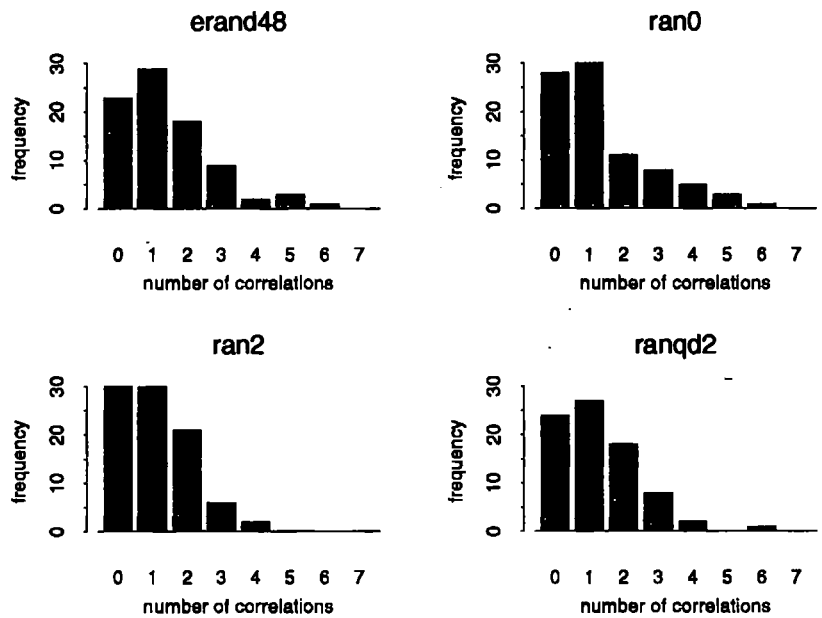
It should be noted that these tests were used as a means of *rejecting* any generator that did not pass rather than as a means of *comparing* one generator with another.

Each of the 100 replications was subjected to serial correlation was test. The autocovariance γ_k for a sequence of random variates u_t, $t = 0, 1, \ldots$, is given

$$\gamma_k = E(u_t - \mu)(u_{t+k} - \mu) \tag{93}$$

where k is the *lag* between random variates. The number of statistically significant (for a significance level of $\alpha = 0.001$) correlations in each replication over $k = 0, 1, \ldots 100$ lags is shown in the graph in Figure 100.

Figure 100 Correlation results

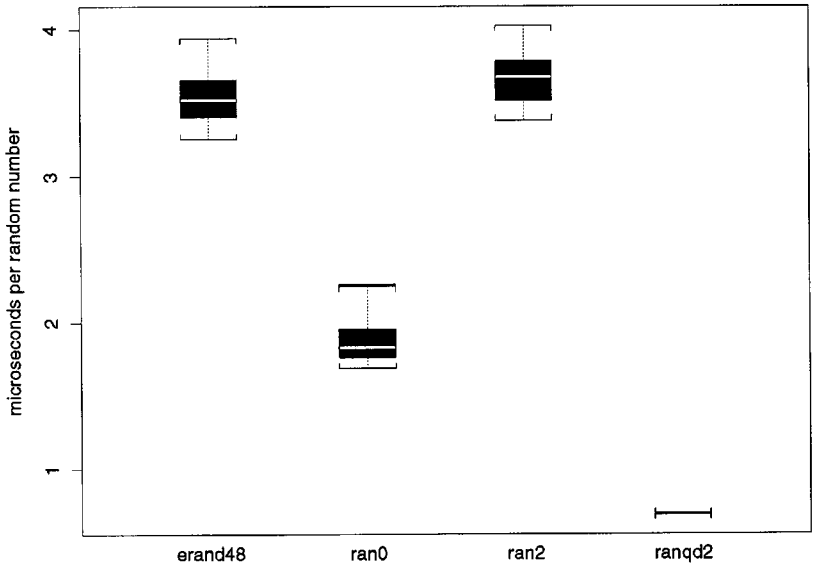


The generator that exhibited the least amount of correlation was ranqd2 while erand48 exhibited the most. However there was not a great deal of difference between the four generators so it is difficult to reject any one generator based on these results.

Performance

As it will be necessary to generate long sequences of random number, generator performance is an issue. Figure 101 shows a graph of the time to generate a random variate from each generator on a Sun SPARC Ultra I with 170MHz processor.

Figure 101 Performance of random number generators



The fastest generator was `ranqd2`, which is not surprising given that it is implemented using only a few lines of code. `ran0` was the next fastest, followed by `erand48` and `ran2`.

Portability

Portability in random generators is the ability to reproduce the same results from machine to machine. `ranqd2` is clearly very machine dependent. The "mod" operation comes from the resulting low-order bits of a 64-bit product on a 32-bit machine.

Portability is affected by availability. `erand48` is an IEEE Posix routine [225] so is only available on POSIX compliant machines. The source code for `ran0` and `ran2` is available from Press et. al. [198] so both these generators can be "ported" onto different platforms.

Period

For Monte Carlo simulations an important feature of a random number generator is its period, that is the number of successive deviates that can be generated before the sequence repeats. A generators *full* period can be no larger than m. Furthermore, inappropriate choices of values of a, c and m can result in a period less than the maximum m, which is undesirable (unless m is very large).

`ran0`, and `ranqd2` are 32 bit generators and each have a modulus of $m = 2^{31} - 1$. `erand48`, while it returns a 32 bit value, it uses 48 bit arithmetic (modulus $m = 2^{48}$). The maximum period of `ran2`, is $>2\times10^{18}$ because it combines two random number generators (L'Ecuyer [143]).

The period of a random number generator is an important issue and determines length of a random number sequence. Press et. al. [198] recommend that the length of a sequence should not exceed 5% of a generators period.

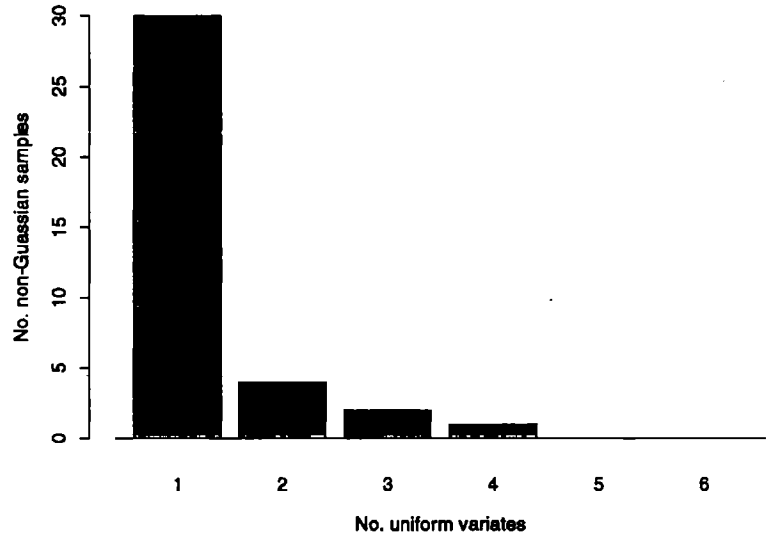
The Proxy server at the Malmesbury site receives in the order of 10,000 request per day. The Monte Carlo simulations of the daily cache growth require about 1000 replications. This requires a sequence of 10^7 Gaussian random numbers. However the generation of each Gaussian variate requires multiple uniform random variates. The convolution method typically requires the most uniform variates per Gaussian variate.

The polar method requires two uniform random variates per Gaussian variate. For the Rejection method the number of uniform variates is variable. From a sample of 30 replications of 10,000 Gaussian variates, the number of uniform random variates (per Gaussian variate) was 3.6.

With the convolution method the greater the number of uniform variates used the closer the resulting variates are to a Normal distribution. The Kolmogorov-Smirnov test was used to test 30 replications of 10,000 variates using $n = 1, ...,6$ uniform variates.

Figure 102 shows the number of samples (out of 30) that were not normally distributed at a significance level of $\alpha = 0.001$.

Figure 102 KS test of Convolution method



According to these results at least 5 uniform random variates need to be generated in order to produce one Gaussian random number. However Jain [123] recommends that 12 uniform random variates should be used.

So we can expect to generate between 2×10^7 and 1.2×10^8 uniform variates, which is between 1.8% and 11.2% of the maximum period of `ran0`, and `ranqd2`!

Selection of Uniform Random Number Generators

All random number generators appear have problems as well as merits. It is therefore a question of finding the most appropriate generator rather than the best one.

Given the results of Chi-square, Kolmogorov-Smirnov, and Serial Correlation tests, none of generators examined appeared to be seriously flawed in terms of statistical compliance.

Furthermore the author is reluctant to reject any of generators based upon the performance results. While `ran0`, and `ranqd2` are faster than `erand48` and `ran2`, the author feels that using the slower generators would not necessarily be a serious bottle neck in time taken to produce ARIMA sequences.

What is a serious concern is the "limited" period of `ran0`, and `ranqd2`. It is quite likely that the number of uniform variates required in the Monte Carlo simulations could exceed 5% of the maximum period of `ran0`, and `ranqd2`. And it is this reason that `ran0`, and `ranqd2` must be rejected in favour of the slower generators.

`ran2` is the more "portable" generator because the source code is available, whereas with `erand48` availability is very much dependent upon IEEE Posix compliance. However, many manufactures support the POSIX standard.

Moreover, even with the generator source code portability may be difficult to achieve. The following program was compiled on a Sun Solaris 2 workstation and a Pentium PC running Windows 95:

```
main()
{
    double x=0.3, y=0.1;
    printf("%g\n", 2-(x-y)/y);
}
```

Despite both compilers being ANSI compliant the output from the Sun workstation was `2.22045e-16`, while the output from the PC was `2.77556e-16`. Clearly double precision arithmetic is not consistent across platforms. This will affect the output of generators from platform to platform.

While `erand48` is not as widely available as `ran2`, it does possess some standards compliance (although, as we have seen, this does not guarantee portability). So `erand48` is the preferred generator out of the four.

Selecting a Gaussian Random Number Generator

Using the `erand48` uniform random number generator, Gaussian random number generators were developed for the Convolution, Polar and Rejection methods. Random variates produced by the generators were tested for "normality" using the Kolmogorov-Smirnov test. The Convolution method was tested for 12 uniform random variates.

Using each method 30 sequences of 10,000 variates were generated and tested. The K-S test revealed no significant deviation from the normal distribution for any of the methods. In conclusion the choice of Gaussian random number generator method would appear to be fairly arbitrary for the application in this thesis. In which case the convolution method is chosen on the basis of simplicity.

Appendix C

During the course of the author's employment with Lucent Technologies there have been many reorganisations. Indeed the company underwent the largest ever corporate reorganisation with the *trivestature* of AT&T in 1996. Many changes to the corporate network have been made in order meet the new business needs.

Many of these changes have taken place during course of this research project which has enabled the author carry out a number of experiments in order to investigate the factors that affect wide area network performance under a variety of conditions. These experiments and the results are described in this chapter.

The evolution of the wide area network (from the UK's point of view) is summarised below:

- 1990: International Bandwidth Management Option (IBMO). Network Systems (a business unit of AT&T) implemented a wide area network for various voice and data applications. The IBMO connected sites in the Holland, the UK and the US.
- 1992: Global Integrated Network (GIN). In order to make more effective use of the global communication resources with the company the IBMO was inte-

grated with wide are networks of the other business units. This resulted in both architectural and capacity changes.

- 1996: Lucent EMEA wide area network. The trivestature of AT&T (into AT&T, NCR and Lucent Technologies) resulted in a disaggregation of the corporate wide area network. The current network serves all Lucent Technologies' business units throughout Europe and Middle East region. Again this resulted architectural and capacity changes.

Effects of Packet size and Bandwidth

The effects of varying packet size and channel bandwidth on packet round trip delays over a wide area network are determined by the following experiment carried out on the GIN network.

Packet round trip delays were measured using PING (Packet INternet Groper) [47]. PING uses an ICMP ECHO_REQUEST packet to elicit an ECHO_REPLY packet from a remote host. The time taken to send an ECHO_REQUEST and receive an ECHO_REPLY constitutes the round trip delay. The PING command was issued from the router in the UK to the remote router in the US. The capacity of the communications channel connecting the two routers was varied by network management personnel in the US. Variation of the channel capacity was possible because this experiment was carried during the commissioning stage of the network. It would be difficult to do this now that the network is in service as it would cause disruption to user service.

A two-factor, full factorial experiment design with replications was employed [123]. Packet size and channel capacity being the two factors investigated here. Packet round trip delays were measured for $a = 6$ levels of packet sizes (expressed in terms of its payload) and $b = 4$ levels of channel bandwidth.

The levels of packet size chosen were: 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1024 bytes and 1472 bytes; and channel bandwidths levels were: 64kb/s, 128kb/s, 256kb/s and 512kb/s.

Furthermore, for each combination of factor levels the round trip delays was measured for $r = 1000$ replications. So the model for packet round trip delay is given by

$$y(ijk) = \eta + x_A(j) + x_B(i) + x_{AB}(ij) + e(ijk) \quad (94)$$

Where

$y(ijk)$ = packet round trip delay for the kth replication with factor A (packet size) at level j and factor B (channel bandwidth) at level i.

η = overall mean round trip delay

$x_A(j)$ = effect of factor A (packet size) at level j.

$x_B(i)$ = effect of factor B (channel bandwidth) at level i.

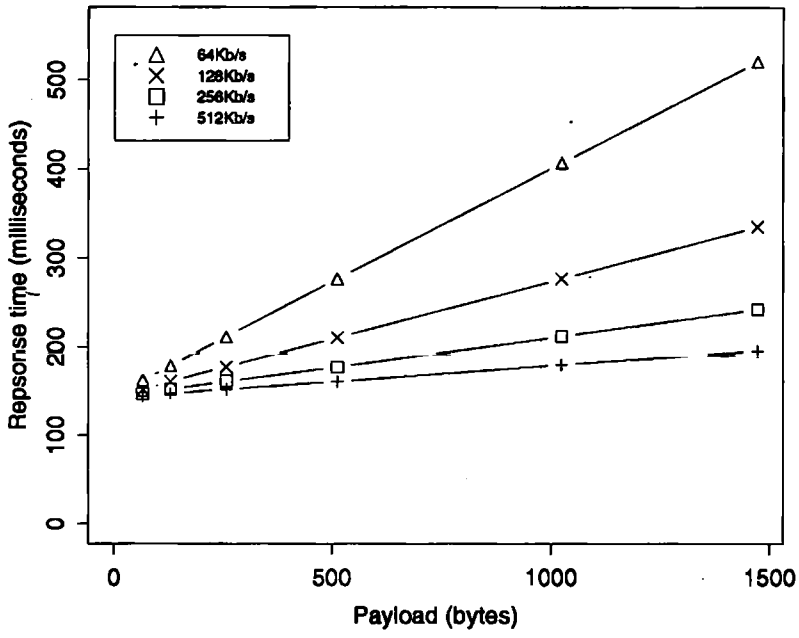
$x_{AB}(ij)$ = effect of the interaction between factor A (packet size) at level j and factor B (channel bandwidth) at level i.

$e(ijk)$ = experimental error.

The effects of each factor (and the interaction) were tested for statistical significance at a significance level of $\alpha = 0.01$ against the F distribution.

The results of this experiment are shown in Figure 103. It can be seen that there was a linear increase (approximately) in the mean round trip delays across packet sizes for all channel bandwidths. The graph also shows that there is an interaction between the two factors.

Figure 103 Mean packet round trip delays



Using the full factorial experimental design described above the allocation of variance of packet round trip delays across packet size channel bandwidth, and the interaction between the factors was determined. Table 14 shows the expressions for the sum of squares of the components.

Table 14 Allocation of variance

Component	Sum of Squares	Percentage of Variation	
y	$\mathrm{SSY} = \sum y_{ijk}^2$		Sum of Squares of y
$\bar{y}_{...}$	$\mathrm{SS0} = abr\eta^2$		Sum of Squares of $\bar{y}_{...}$
$y - \bar{y}_{...}$	$\mathrm{SST} = \mathrm{SSY} - SS0$		Total Sum of Squares
A	$\mathrm{SSA} = br\sum x_A(j)^2$	$100 \times \left(\frac{\mathrm{SSA}}{\mathrm{SST}}\right)$	Sum of Squares of $x_A(j)$
B	$\mathrm{SSB} = ar\sum x_B(i)^2$	$100 \times \left(\frac{\mathrm{SSB}}{\mathrm{SST}}\right)$	Sum of Squares of $x_B(i)$
AB	$\mathrm{SSAB} = abr\sum x_{AB}(ij)^2$	$100 \times \left(\frac{\mathrm{SSAB}}{\mathrm{SST}}\right)$	Sum of Squares of $x_{AB}(ij)$
e	$\mathrm{SSE} = SST - (\mathrm{SSA} + \mathrm{SSB} + \mathrm{SSAB})$	$100 \times \left(\frac{\mathrm{SSE}}{\mathrm{SST}}\right)$	Sum of Squared Errors

Table 15 shows the proportion of the total variation of the round trip delay allocated to the effects of packet size (A), channel bandwidth (B) and the interaction between the two factors (AB).

Table 15 Allocation of variance results

Component	Sum of Squares	Sum of Square Values	Percentage of Variation
y	SSY	1,305,146,529.00	
$\bar{y}_{...}$	SS0	1,101,600,288.00	
$y - \bar{y}_{...}$	SST	203,546,301.00	
A	SSA	93,562,283.00	45.97
B	SSB	93,562,283.00	28.99
AB	SSAB	44,166,704.00	21.70
e	SSE	6,815,993.00	3.35

The effect of the packet size explained 45.97% of the total variation in round trip delay and the effect of channel bandwidth explained 28.99%. 21.70% was due to the interaction. The proportion of variation due to experimental error was only 3.35%

Table 16 shows the expressions for testing statistical significance of the variance due to the effects of packet size, channel bandwidth and the interaction between the two factors. The ratios MSA/MSE, MSB/MSE and MSAB/MSE, have F distributions.

Table 16 Analysis of variance

Component	Mean Squares	F-Component	F-Table
A	$\text{MSA} = \frac{\text{SSA}}{(a-1)}$	$\frac{\text{MSA}}{\text{MSE}}$	$F_{[1-\alpha;(a-1),ab(r-1)]}$
B	$\text{MSB} = \frac{\text{SSB}}{(b-1)}$	$\frac{\text{MSB}}{\text{MSE}}$	$F_{[1-\alpha;(b-1),ab(r-1)]}$
AB	$\text{MSAB} = \frac{\text{SSAB}}{(a-1)(b-1)}$	$\frac{\text{MSAB}}{\text{MSE}}$	$F_{[1-\alpha;(a-1)(b-1),ab(r-1)]}$
e	$\text{MSE} = \frac{\text{SSE}}{ab(r-1)}$		

Where $a-1$, $b-1$, $(a-1)(b-1)$ and $ab(r-1)$ are the degrees of freedom for SSA, SSB, SSAB and SSE respectively.

The analysis of variance results are shown in Table 17. The F-ratios are higher than the values from the F distribution table [123], therefore all three effects are statistically significant for $\alpha = 0.01$.

Table 17 Analysis of variance results

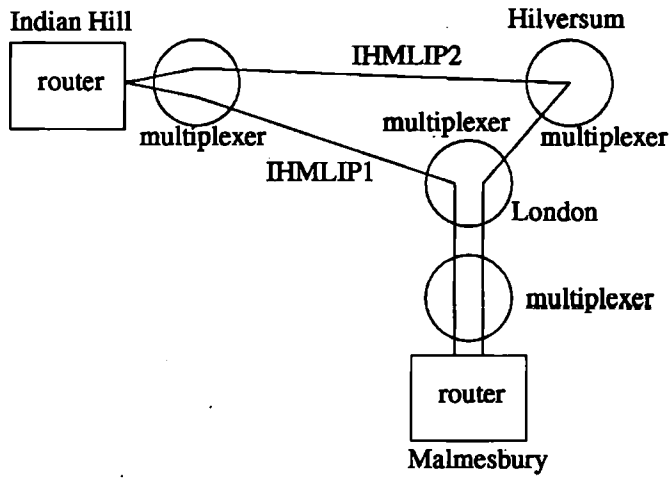
Component	Degree of Freedom	Mean Square	F-Component	F-Table
A	5	18,712,456.67	65,823.11	$F_{[0.99;5,23976]} = 3.05$
B	3	19,667,106.96	69,181.20	$F_{[0.99;3,23976]} = 3.82$
AB	15	2,944,446.92	10,357.41	$F_{[0.99;15,23976]} = 2.08$
e	23,976	284.28		

It has been shown that, within the range of the factor levels examined here, packet size has more of an effect on round trip delay than channel capacity. When response times are slow the tendency is to buy a faster serial link. However increasing channel capacities may not necessarily bring about the best improvements in performance (and can come at a high price). In which case it may be necessary to look to compression techniques as a means of reducing packet sizes and therefore improving round trip delays.

Propagation Delays

The IBMO network was implemented so that developers in the UK could work on systems at the Bell Laboratories facility based in Indian Hill. Due to the critical nature of the work done by developers it was necessary to have two 224kb/s circuits (referred to as IHMLIP1 and IHMLIP2) to ensure high availability of the network. Furthermore, in order to ensure diversity, IHMLIP2 was routed via another Lucent Technologies site at Hilversum in Holland so that it used a different transatlantic facility than IHMLIP1 which was routed via a site in London. Figure 104 shows the routed taken by IHMLIP1 and IHMLIP2.

Figure 104 IBMO UK to US circuits.



The packet round trip delays over these circuits were measured using PING. It can be seen from the graph in Figure 105 that round trip delays over IHMLIP2 are higher than for IHMLIP1 which is due extra distance covered by IHMLIP2 as a result of being routed through Holland.

A linear regression model for the packet round trip delay D_{RTT1} over the IHMLIP1 channel is given by

$$D_{\text{RTT1}} = 9.5\times10^{-6} \times \text{packet\_size} + 0.11 \quad (95)$$

where delay is in seconds and packet size is in bits. Dividing (95) through by 2 gives the one-way packet latency D_{latency1} for IHMLIP1

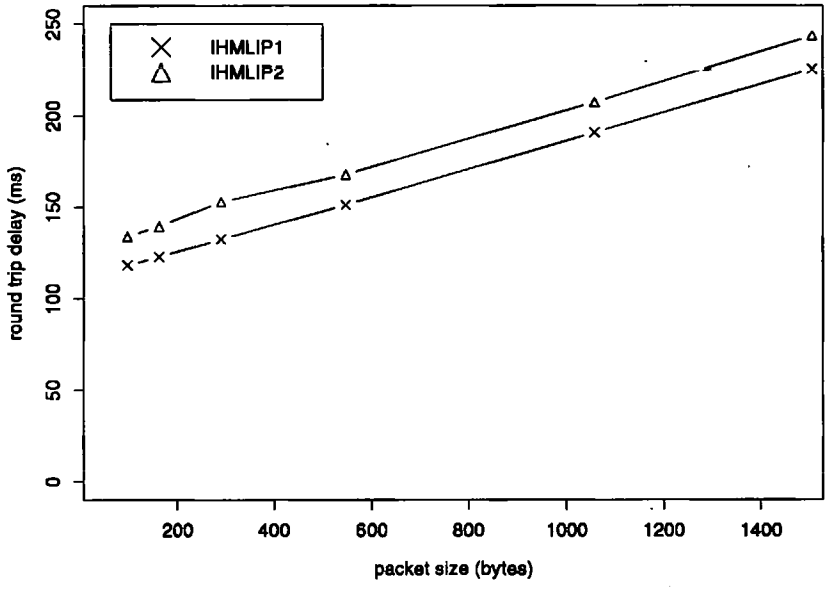
$$D_{\text{latency1}} = 4.8\times10^{-6} \times \text{packet\_size} + 0.055 \quad (96)$$

Similarly, a round trip delay model can be derived for IHMLIP2 and from it a model for one-way packet latency D_{latency2}

$$D_{\text{latency2}} = 4.8\times10^{-6} \times \text{packet\_size} + 0.065 \quad (97)$$

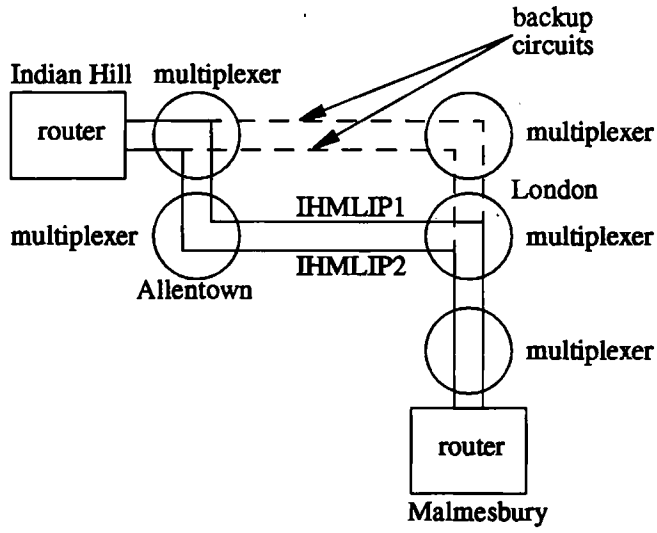
If the packet_size is set to zero D_{latency1} and D_{latency2} gives the signal propagation delays (D_p in equation (8) in CHAPTER 2) for IHMLIP1 and IHMLIP2 respectively. It can be seen that the signal propagation delay for IHMLP1 is 55ms, whereas IHMLIP2 is 65ms.

Figure 105 UK to US packet round trip delays



The effects of distance on packet round trip delay can be further illustrated by comparing these results with the results in the previous section. When the IBMO was replaced by the GIN, the network underwent major architectural changes. Figure 106 shows the routes taken by IHMLIP1 and IHMLIP2 circuits. While there was still a need for high network availability, circuits did not have to take diverse routes because the multiplexers were able to dynamically *reroute* them over a different path (the IBMO multiplexers did not have this functionality) in the event of a failure of a facility. The multiplexers computed the optimum path for the circuits based upon the least number of (multiplexer) hops. During normal working conditions IHMLIP1 and IHMLIP2 were routed through a site in Allentown (four multiplexer hops).

Figure 106 GIN UK to US circuits

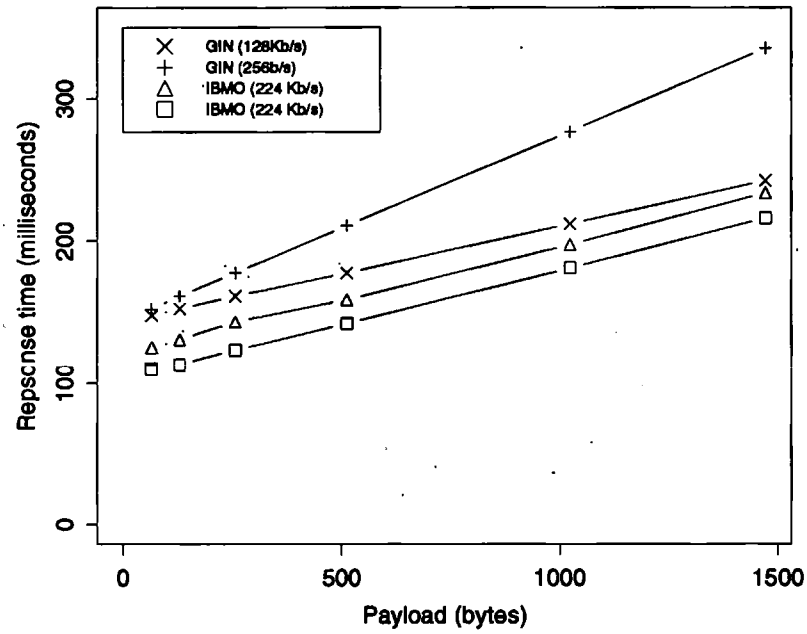


The experiment in the previous section was carried out in this environment. Unfortunately the GIN network could only provide channel capacities for multiples of 64kb/s (whereas IBMO could clock router interfaces at arbitrary speeds) so the packet round trip delay results are not entirely comparable between the two networks. Despite this the effects of propagation delays are still apparent.

The graph in Figure 107 shows the round trip delays over the two IBMO circuits (IHMLIP1 and IHMLIP2). The graph also shows the round trip delays for IHMLIP1 at 128kb/s circuit and a 256kb/s circuit over the GIN network.

One would expect that round trip delays over the 128kb/s GIN circuit to be higher than over the 224kb/s IBMO circuits. However, while round trip delays improved when the GIN circuit was upgraded to 256kb/s, the delays were still higher than the lower speed IBMO circuits (224kb/s).

Figure 107 Packet round trip delays, IBMO versus GIN



The architectural changes in the network meant that this circuit between Malmesbury and Indian Hill was routed via a Lucent Technologies site in Allentown which is about 1,000 miles south of Indian Hill.

Figure 108 Current Lucent backbone network

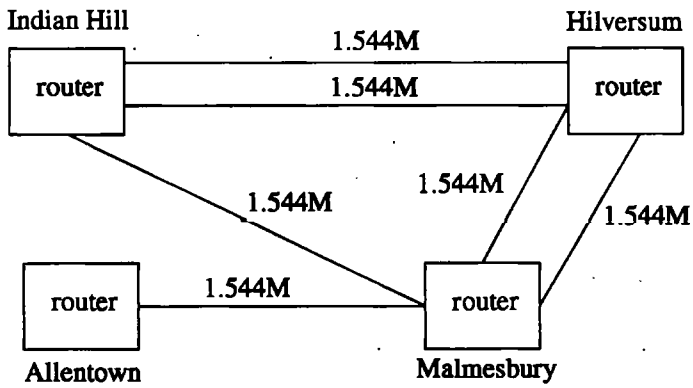
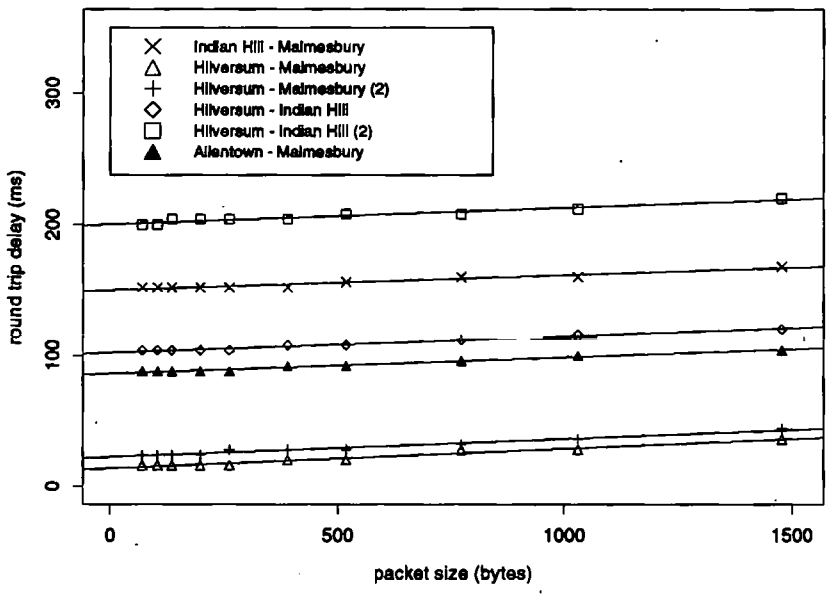


Figure 109 Packet delays in the current Lucent network



Ping was used to sample round trip delays across each circuit. A regression model for packet latency derived from this data.

$$D_{\text{latency}} = a \times \text{packet\_size} + b \quad (98)$$

Table 18 shows the values for the parameters a and b for each link.

Table 18 Packet latency regression model parameters

Circuit	$a \times 10^{-6}$	$b \times 10^{-3}$
Indian Hill - Malmesbury	0.70	75
Hilversum - Malmesbury (1)	0.95	7
Hilversum - Malmesbury (2)	0.85	11
Hilversum - Indian Hill (1)	0.75	51
Hilversum - Indian Hill (2)	0.80	100
Allentown - Malmesbury	0.75	43

One of the most notable results is the difference between the two Hilversum-Indian Hill circuits. The signal propagation delay of one is nearly double the other. The high propa-

gation delay of Hilversum - Indian Hill (2) is a "managed access" circuit (whereas Hilversum - Indian Hill (1) is not). Managed access circuits are circuits that can be dynamically rerouted within the PTT/Telco network in the event of a failure. While this increases circuit availability it would appear there is an incurred delay cost. It was the belief of the Lucent network architects that the reason for this delay cost was the additional network hardware required to provide managed access services. At the time of writing this thesis there were plans to remove this circuit from managed access. The added availability was not deemed necessary (considering the delay costs) due to current level of redundancy already in the Lucent network.